INTERNET-DRAFT Soobok Lee
draft-ietf-idn-lsb-ace-01.txt
Expires 2002-Jan-03 2001-Jul-03

Improving ACE using code point reordering v1.0

Status of this Memo

Abstract

This document describes code point reordering to improve ACE compression algorithms. Being based on character frequency and character adjacency statistics for major characater sets, this reordering can be easily implemented only with simple character mapping tables without adding complexity to existing ACE algorithms.

When applied to DUDE and AMC-ACE-W, this reordering greatly improves both ACEs' compression ratios for Hangul, Chinese, Vietnamese, Katakana and European domains. Interestingly, reordered DUDE shows better or equal compression ratio than both bare AMC-ACE-W and reordered AMC-ACE-W.

Contents

Differences from version 0.9

version 1.0 differs from version 0.9 in four respects:

1) For Hangul,
   it does not use Hangul jamo frequency order any more:
   instead, it adopts new reorderding based on both
   hangul character frequency and adjacency in words
   used as business names in Korea.

2) For Unified Han, like for Hangul,
   it adopts new reorderding table that reflects
   han character frequency and adjacency in words
   used as business names in China/Taiwan/Japan.

3) new supports for Japanese katakana

4) additional implementation of reordering with AMC-ACE-W

Overview

Pursuing shorter ACE labels is justified to save memory resources
and to reduce internet traffic even for domains of average length
in various application/core internet protocols.

Both 11172 Hangul syllables and 24000 or more CJK Han syllables
occupy roughly half of the entire unicode space.
Their lexicographical ordering(not in frequency ordering) makes
various ACE compression technique work poorly for them, because.
they are spread evenly through out those wide ranges.

The most frequent 256 Hangul syllables has cumulative frequency sum
of 88.2% and for the case of top 512 ones , it reaches 99.9%.

The most frequent 256 Han letters has cumulative frequency sum
of 58.2% and for the cases of top 512,1024,2048 and 4096 ones,

it reaches 72.8,85.9,95.4 and 99.4%, respectively.

Even, Latin characters code range, including 'a' - 'z' has lexicographical order that does not reflect the fact that 's','t' and 'r' are more frequently used than 'j','k' and 'h'.

Most ACE algorithms show good compression ratio when frequently used characters are located in narrow code ranges.
Especially, to reduce DUDE XOR distance, we can make the narrow area fit in aligned blocks of 16,256 or 4096 code points.

Unified Han and Hangul

Most frequently used 4096 Traditional Chinese/Simplified Chinese/ Japanese Kanji letters are reordered into single aligned block of 4096 code points. Their combinations are estimated to form almost 99% of modern chinese business names.

Most frequently used 888 Hangeul syllables are reordered into the lower portion of single aligned block of 4096 code points.
Their combinations are expected to cover almost 99% of modern hangul business names.

In fact, every block of 256 code points in these reordered areas is designed to reflect not only character frequency order, but also to reflect adjacency preference derived from statistics on major business category names or famous regional names in eastern asia.

For example, there is a frequent korean industrial category name 'jeon-ja' (electronics). In pure frequent-oriented ordering, its two component hangul syllables 'jeon' and 'ja' should have been put far apart from each other. But in this new adjacency-adjusted frequency ordering, they are put together in a single row (u+???0 ~ u+???f) in order to reduce the XOR distance toward single quintet(for DUDE).

The han/hangul frequency mapping tables and its statistical data are constructed from business names found in internet directory sites at {cn|tw|kr}.yahoo.com.

Japanese Katakana and Hiragana

Japaneses hiragana (u+3040 ~ u+309f) shows relatively even frequency distribution in japanese business names. And it is often replaced with its Kanji (Japanese Han letter) equivalent in registerd business names.
I have no mapping table for hiragana, yet.

Japaneses katakana (u+30a0 ~ u+30ff) has been widely used to express foreign or english words in Japanese. Most frequent 10 katakanas'

cumulative frequency is estimated to be around 40%.

The katakana frequency mapping table is constructed from business names found in internet directories sites such as www.yahoo.co.jp.

Basic Latin

Basic Latin row u+0070 ~ u+007f has 'p','r','s','t' and 'u' which are more frequently used in European nouns than '`','j','k','f' and 'g' in u+0060 ~ u+006f row which includes most frequently used 'a'~'o' .

If these two sets of 5 characters are swapped character-wise, 'p','r','s','t','u' go into the u+0060 ~ u+006f row.

Any character sequence only from this single aligned block of 16 codes has XOR-distance or code window length shorter than 0x10 and makes DUDE and other ACEs do good compression.

Extended Latin and Combining Diacritical Marks

First 6 rows from Latin Extension A(u+0100 ~ u+015f) and 6 rows from Basic Latin & Latin-1 Supplement (u+0000 ~ u+002f and u+0080 ~ u+00a0) are swapped.
First 3 rows from Combining Diacritical Marks(u+0300 ~ u+032f) and 3 rows from Latin-1 Supplement (u+00B0 ~ u+00df) are also swapped.

This makes frequently used parts of Latin Extended-A and Combining Diacritical Marks go into first align block of 256 codes points (u+0000 ~ u+00ff). Any character sequences from this single block make XOR-distance or code window length much shorter than 0x100.

This improvement benefits especially East-European and Vietnamese that use Latin Extented A and Combining Diacritical Marks.

Other character sets

For Arabic,Cyrillic and Hindi etc, we can devise similiar frequency mapping tables as that for katakana.

Modified Encoding procedure of DUDE implementation of this idea

All ordering of nybbles and quintets is big-endian (most significant first). A nybble is 4 bits. XOR is bitwise exclusive or.

```
    This modification is hyphen-safe.
    Hyphen encoding and decoding are not affected by this modification.

    let prev = 96
    for each input integer n (in order) do begin
      if n == 45 then output hyphen minus
      else begin

        n = reorder(n)   //  ******** ADDED **********

        let diff = prev XOR n
        extract the least significant nybbles of diff, as few as are
          sufficient to hold all the nonzero bits (but at least one)
        prepend 0 to the last nybble and 1 to the rest
        output base-32 characters corresponding to the quintets
        let prev = n
      end
    end

    The encoder must either correctly handle all integer values that can
    be represented in the type of its input, or it must check whether
    the input contains values that it cannot handle and return an error
    if so.  Under no circumstances may it produce incorrect output.


Modified Decoding procedure of DUDE implementation of this idea

    let prev = 96
    while the input string is not exhausted do begin
      if the next character is hyphen-minus then output 45
      else begin
        input characters and convert them to quintets until
          encountering a quintet beginning with 0
        fail upon encountering a non-base-32 character or end-of-input
        strip the first bit of each quintet
        concatenate the resulting nybbles to form diff
        let prev = prev XOR diff

        output restore_order(prev)  //  ******** MODIFIED **********

      end
    end
    encode the output sequence and compare it to the input string
    fail if they are not equal


Modified Encoding and decoding algorithms of AMC-ACE-W for this idea

    (This modification does not affect literal mode of AMC-ACE-W).
    procedure initialize(refpoint,style,literal):
      let refpoint[1..5] = (0xE0, 0xA0, 0, 0, 0x10000)
      let style = 0
```

```
  let literal = false

procedure update(refpoint,style,n,k):
  # Update the active style and reference points based on
  # the latest code point (n) and the number of base-32
  # characters used to represent it (k).
  let style = k < 3 ? 0 : k > 3 ? 1 : style
  let refpoint[1] = (n >> 4) << 4
  if (k > 2) then let refpoint[2] =
    n is in 00A0..017F ? 0xA0 : (n >> 8) << 8
  if (k > 3) then let refpoint[3] = n is in 3000..9FFF ? 0x4E00 :
    style == 1 and n is in 0xA000..0xD7FF ? 0x8800 : (n >> 12) << 12

procedure encode:
  constant maxdelta[0][1..5] = (0xF, 0xFF,  0xFFF, 0xFFFF, 0xFFFFF)
  constant maxdelta[1][2..5] = (     0xFF, 0x4FFF, 0xFFFF, 0xFFFFF)
  initialize(refpoint,style,literal)
  for each input code point n (in order) do begin
    # Check code point range to avoid array bounds errors later:
    if n is not in 0..10FFFF then fail
    if n == 0x2D then output two hyphen-minuses
    else if n represents an LDH character then begin
      # Letter/digit is encoded literally, so get into literal mode.
      if not literal then output hyphen-minus
      let literal = true
      output the character represented by n
    end
    else begin
      # Non-LDH code point is encoded in base-32.
      # Compute the number of base-32 characters to use:

      n = reorder(n)  // ADDED ************************

      for k = 1 + style to infinity do begin
        let delta = n - refpoint[k]
        if delta is in 0..maxdelta[style][k] then break
      end
      # Switch to base-32 mode if necessary:
      if literal then output hyphen-minus
      let literal = false
      # Check for the extended delta of style 1 window 3:
      if k == 3 and delta >= 0x1000
      then represent (delta - 0x1000) in base 32 as three quintets
      else begin
        # Normal case, four bits per quintet:
        represent delta in base 16 as k quartets
        prepend 0 to the last quartet and 1 to each of the others
      end
      output a base-32 character corresponding to each quintet
      update(refpoint,style,n,k)
    end
```

```
      end

    procedure decode:
      initialize(refpoint,style,literal)
      while the input string is not exhausted do begin
        read the next character into c
        # Unpaired hyphen-minus toggles the mode:
        if c is hyphen-minus and the next character is not
        then read the next character into c and toggle literal
        # Double hyphen-minus represents 0x2D:
        if c is hyphen-minus
        then read the next character and append 0x2D to history
        else if literal then append the code point of c to history
        else begin
          # Decode a base-32 sequence.
          convert c to a quintet
          while a quintet beginning with 0 has not been seen
          do read and convert up to four more characters
          concatenate the lowest four bits of each quintet to form delta
          # Check for the extended delta of style 1 window 3:
          if style == 1 and there was only one quintet then begin
            read two characters and convert them to two more quintets
            concatenate delta and the two quintets to form a new delta
            let delta = delta + 0x1000
          end
          let k = the number of quintets decoded
          let n = refpoint[k] + delta
          update(n,k)

          output restore_order(n)  // MODIFIED *****************

        end
      end
      # Enforce the uniqueness of the encoding:
      encode the output sequence and compare it to the input string
      fail if they are not equal
```

Example strings

About 30%~58% improvement in DUDE compression ratio is achieved in these Hangul examples.

LDUDE and LAMCW denote reordering-applied DUDE-02 and AMC-ACE-W, respectively. (AMCW for AMC-ACE-W).
Most examples show LDUDE outperforms LAMCW.

(K1) Korean String 1: (24 hangul syllables)
  u+C138 u+ACC4 u+C758 u+BAA8 u+B4E0 u+C0AC u+B78C u+B4E4
  u+C774 u+D55C u+AD6D u+C5B4 u+B97C u+C774 u+D574 u+D55C

```
  u+B2E4 u+BA74 u+C5BC u+B9C8 u+B098 u+C88B

  DUDE-02 : 6txiy79ny53nz79a8wizwwnzzuavyizv3atuuiz2vby27jz66iz8sit\
            usauiyz5i23az96iz6ze3xaz2td ( 82 chars )
  LDUDE   : 5suhxb9jt2pydtwetwkxhtsrxhbyhvsmvvk7r2ityd6atqt8etvittk
            ( 55 chars, 33.9% shorter )
  AMCW    : 6tvifgem42ixihhakfnh6nhhem5wrk6fmpmpwim6m5wrmwxn5u8eivw\
            mp6iqige2nem ( 67 chars )
  LAMCW   : 5swhtg8r5tycsb5swfgirxi5sxhsabyg5vypgcz2isa5tyd4d5p5sxj\
            gmbgd5 ( 61 chars )


(K2) Korean String 2:  ( 9 hangul syllables )
  <KRNIC in korean>
  U+D55C U+AD6D U+C778 U+D130 U+B137 U+C815 U+BCF4 U+C13C
  U+D130

  DUDE-02 : 7xvNz2vBy4tFtywIyssHz3uCzw8Bz76ItssN
            ( 36 chars )
  LDUDE   : 5syAB3BIJ7BB7NF
            ( 15 chars,  58.3% shorter )
  AMCW    : 7xxNFmpM52QjsGjzNaxJhwKj6Qjs
            ( 28 chars )
  LAMCW   : 5ssAsB3AIBwAB3PI
            ( 16 chars )


(K3) Korean String 3:  ( 18 hangul syllables )
  U+C804 U+AD6D U+C2E4 U+C9C1 U+B178 U+C219 U+C790 U+B300
  U+CC45 U+C885 U+AD50 U+C2DC U+BBFC U+B2E8 U+CCB4 U+D611
  U+C758 U+D68C

  DUDE-02 : 62yEyxyJy92J5uFz25JzvyBx2Jzw3Az9wFw6Ayx7Fy92Nz3uA3tEz8\
            xNt44FttwJtt7E ( 68 chars )
  LDUDE   : 5szAtBtvBt7Mt2Qv4Qu7KtFt5It3MuEvAtvDyJCtuC4G4J
            ( 46 chars,  32% shorter )
  AMCW    : 62sEFmpKzeNqbGm2Ks3M6sG2aPcfNefFksKy6I96GziPfwRstM42Rwn
            ( 55 chars )
  LAMCW   : 5stAsB5tvAGhmGmgG2mGatsE5t7JGbhsDvD5tsAyIK5swJ8RwG
            ( 50 chars )


(K4) Korean String 4:  ( 7 hangul syllables )
  <Hynics Semiconductor in korean>
  U+D558 U+C774 U+B2C9 U+C2A4 U+BC18 U+B3C4 U+CCB4

  DUDE-02 : 7xvItuuNzx5PzsyPz85N97Nz9zA
            ( 27 chars )
  LDUDE   : 5s3C4F5Q7PtwRtMK
            ( 16 chars,  40% shorter )
  AMCW    : 7xxIM5wGyjKxeJa2G8ePfw
            ( 22 chars )
```

```
  LAMCW   : 5s9CxH8JvE5tzMyAK
            ( 17 chars )


(K5) Korean String 5:  ( 13 hangul syllables )
  U+D658 U+ACBD U+C6B4 U+B3D9 U+C5F0 U+D569 U+BC18 U+D575
  U+D2B9 U+BCC4 U+C704 U+C6D0 U+D68C

  DUDE-02 : 7yvIz48Fy4sJzxyPzyuJts3Jy3zBy3yPz6Ny8zPz56At7EtsxN
            ( 50 chars )
  LDUDE   : 5s7NB4EDvHFtxDv5Kv6NtIt4R5GwK
            ( 29 chars, 42% shorter )
  AMCW    : 7yxIFf7MxwG83MrsRmjJa2RmxQx3JgeM2eMysRwn
            ( 40 chars )
  LAMCW   : 5s5N5PtJKuPI5tzMGybGiptF5s5KsNwG
            ( 32 chars )



About 35%~50% improvement in DUDE compression ratio is achieved in
these UniHan examples.

(TC1) Traditional Chinese String 1: ( 16 letters )
  u+5354 u+91c7 u+5065 u+5eb7 u+4e8b u+696d u+670d u+52d9
  u+7db2 u+002d u+5354 u+91c7 u+6709 u+9650 u+516c u+53f8

  DUDE-02 : xvve6u3d6t4c87ctsvnuz8g8yavx7eu9ym-u88g6u3d9y6q9txj6z\
            vnu3e  ( 58 chars)
  LDUDE   : xs8qy7ny9jhyi6f6bb8h-4iy7nyxkbed
            ( 32 chars, 44.8% shorter)
  AMCW    : xvxen8huyfafzs2mc5pcipw7jh7u--xxen8hcijqcsvynx9i
            ( 48 chars )
  LAMCW   : xs2q2xcu4m4n6esb6abug--2q2xcusijpq
            ( 34 chars )


(TC2) Traditional Chinese String 2: ( 21 letters )
  u+5317 u+4eac u+5e02 u+91ab u+85e5 u+7d93 u+6fdf u+6280
  u+8853 u+7d93 u+71df u+516c u+53f8 u+5fa1 u+91ab u+7db2
  u+7d61 u+83ef u+91ab u+7db2 u+8def

  DUDE-02 : xvzht75mts4q694jtwwq92zgtuwn7xr847d9x6a6wnus5du3e6xj6\
            8sk86tj7d982qtuwe86tj9sxp ( 78 chars)
  LDUDE   : xtwicfz6b99a38g27c2vdd8cz7mzuqdt6izuiy6iz5nz5fy6by6ib
            ( 53 chars, 32.0% shorter)
  AMCW    : xvths4naacn7mj9fh6veq9beakuvh6ve89vynx9iapbn7mh7uyb2v\
            8rn7mh7um9r ( 64 chars )
  LAMCW   : xtuiukr28q5tqu9i4ukutjk9i3uduspqv6g28quug33kuur28quugh
            ( 54 chars )


(TC3) Traditional Chinese String 3: ( 18 letters )
```

```
  u+795e u+8fb2 u+7db2 u+990a u+8eab u+4fdd u+5065 u+7db2
  u+5065 u+5eb7 u+4e16 u+754c u+5065 u+5eb7 u+8a2d u+8a08
  u+5bb6 u+60e0

  DUDE-02 : z3vq9y8n9usa8w5itz4b6tzgt95iu77hu77h87cts4bv5xkuxuj87\
            c7w3kuf7t5qv5xg ( 68 chars )
  LDUDE   : xwsiw5e9kzyqz8fhb2p2phtvgxtbwuah8qbtwmyg
            ( 40 chars, 41.1% shorter )
  AMCW    : z3xqnpuh7uq2knfmt7puyfh7uuyfafzstgf4nuyfafzmbpsi75gys\
            8a ( 55 chars )
  LAMCW   : xwyiu7nug3wiu4pkmug4mnv3ky2mu4mnwcdvsiyq
            ( 40 chars )


(SC1) Simplified Chinese String  1 : ( 16 letters )
  <ministry of foreign trade and economic cooperation, PRC>
  u+4e2d u+534e u+4eba u+6c11 u+5171 u+548c u+56fd u+5bf9
  u+5916 u+8d38 u+6613 u+7ecf u+6d4e u+5408 u+4f5c u+90e8

  DUDE-02 : w8wpt7ydt79euu4mv7yax9puzb7seu8r7wuq85umt27ntv2bv3wgt\
            5xe795e ( 60 chars )
  LDUDE   : xswjuzru6nu7fv7kv4gutrwgb7mbwiu6cuzqqxm
            ( 39 chars, 35.0% shorter )
  AMCW    : w8up29ps5kdst5uh7ygsup29pm3cb39n8tknpb39hkygswhdysupa\
            qd ( 55 chars )
  LAMCW   : xsujwxgu3kwwrv3fwvduunykm5ab9jwvmuwfmta
            ( 39 chars )


(SC2) Simplified Chinese String  2 : ( 18 letters )
  u+4e2d u+56fd u+4eba u+6c11 u+5927 u+5b66 u+4e2d u+56fd
  u+8d22 u+653f u+91d1 u+878d u+653f u+7b56 u+7814 u+7a76
  u+4e2d u+5fc3

  DUDE-02 : w8wpt27at2whuu4mvxvguwbtxwmt27a757r82tp9w8qtyxn8u5ct\
            8yjvwcuycvwxmtt8q ( 69 chars )
  LDUDE   : xswjf5gu7fu6rb4ifz8dx6ju8gnu8kwugy8fd8rd
            ( 40 chars, 42.2% shorter )
  AMCW    : w8up29ps5kdst5uh7ygsup29pm3cb39n8tknpb39hkygswhdysupaqd
            ( 55 chars )
  LAMCW   : xsujun3kwwru2abujn36rwsgu8anwsg2uau6fgujk
            ( 41 chars )


About 20%~35% improvement in DUDE compression ratio is achieved in
these Japanese Kanji/Katakana examples.

(JP1) Japanese String 1: ( 25 letters )
  U+793E U+56E3 U+6CD5 U+4EBA U+65E5 U+672C U+30CD U+30C3 U+30C8
  U+30EF U+30FC U+30AF U+30A4 U+30F3 U+30D5 U+30A9 U+30E1 U+30FC
  U+30B7 U+30E7 U+30F3 U+30BB U+30F3 U+30BF U+30FC
```

```
  DUDE-02 : z3xQu97Pv4vGuuyRu5xRu6Jxz8BQMuHtDxDMxHuGzNwItPwMxAtE\
            wIwIwNwD  (60 chars)
  LDUDE   : xs8Nu2Cu4RvMGBysxGyCKtHtQCPFtAyPyKtPBGPyAyAyFyR
            ( 47 chars, 21.6% shorter)
  AMCW    : z3vQ28DDyxs5KB9fCjnvs6P6DI8R9N4RE9D7F4J8B9N5H8H9D5M9\
            D5R9N ( 57 chars )
  LAMCW   : xs2NwsQu4B3KNPvs6M4JD5E4KIFA5A7P5H4KMPA6A4A6F4K
            ( 47 chars )


(JP2) Japanese String 2: ( 15 letters )
  U+8CA1 U+56E3 U+6CD5 U+4EBA U+5317 U+6D77 U+9053 U+81EA
  U+7136 U+4FDD U+8B77 U+63A8 U+9032 U+5354 U+4F1A

  DUDE-02 : 266B74wCv4vGuuyRt74Pv8yA97uEtt5J9s7Nv88M6w4K827R9v3K\
            6vyGt6wQ (60 chars)
  LDUDE   : xs3Hu9Ju4RvMt5CFvuGvsRxtGw5Iz2Ev6BzIwtJE
            ( 40 chars, 33.3% shorter)
  AMCW    : 264B28DDyxs5KxtHD5zNuvI9kE3yt7PMmzBpiNtuxxEttK
            ( 46 chars )
  LAMCW   : xs9HwsQu4B3KvuIPwsMvsEytCu4K3uQy8R3Hu2QK
            ( 40 chars )


(JP3) Japanese String 3: ( 17 letters )
  U+6771 U+4EAC U+90FD U+60C5 U+5831 U+30B5 U+30FC U+30D3 U+30B9
  U+7523 U+696D U+5065 U+5EB7 U+4FDD U+967A U+7D44 U+5408

  DUDE-02 : yztBu37P78xB9svIv29Ey22EwJuRyKwx3Kt6wQv3sI87CttyK734\
            H85vQu3wN (61 chars)
  LDUDE   : xttHxPvtFu9CDyssAyEyHyRys9PxQ4KHGEu4CuwJ
            ( 40 chars, 34.4% shorter)
  AMCW    : z3vQ28DDyxs5KB9fCjnvs6P6DI8R9N4RE9D7F4J8B9N5H8H9D5M9\
            D5R9N ( 57 chars )
  LAMCW   : xs2NwsQu4B3KNPvs6M4JD5E4KIFA5A7P5H4KMPA6A4A6F4K
            ( 47 chars )



LDUDE also shows the same good compression ratio for Latin family of
scripts.

(L1) Vietnamese: ( 38 syllables using diacritical marks )
  Ta<dotbelow>isaoho<dotbelow>kh<ocirc>ngth<ecirc><hookabove>chi\
  <hookabove>no<acute>iti<ecirc><acute>ngVi<ecirc><dotbelow>t
  U+0054 u+0061 u+0323 u+0069 u+0073 u+0061 u+006F u+0068 u+006F
  u+0323 u+006B u+0068 u+00F4 u+006E u+0067 u+0074 u+0068 u+00EA
  u+0309 u+0063 u+0068 u+0069 u+0309 u+006E u+006F u+0301 u+0069
  u+0074 u+0069 u+00EA u+0301 u+006E u+0067 U+0056 u+0069 u+00EA
  u+0323 u+0074

  DUDE-02 : vEvfvwcvwktktcqhhvwnvwid3n3kjtdtn2cv8dvykmbvyavyhbvyqv\
```

```
            yitptp2dv8mvyrjvBvr2dv6jvxh ( 82 chars )
  LDUDE   : uGuh5c5kckqhh5n4atm3n3ktmtdq2cxd7kmb7a7hb7q7irr2dxm7rt\
            muDvr2dvj5f (66 chars , 16 chars(19%) shorter)
```

```
(L2) Spanish: ( using basic Latin & Latin Supplement )
  Porqu<eacute>nopuedensimplementehablarenEspa<ntilde>ol
  U+0050 u+006F u+0072 u+0071 u+0075 u+00E9 u+006E u+006F u+0070
  u+0075 u+0065 u+0064 u+0065 u+006E u+0073 u+0069 u+006D u+0070
  u+006C u+0065 u+006D u+0065 u+006E u+0074 u+0065 u+0068 u+0061
  u+0062 u+006C u+0061 u+0072 u+0065 u+006E U+0045 u+0073 u+0070
  u+0061 u+00F1 u+006F u+006C

  DUDE-02 : vAvrtpde3n2hbtrftabbmtptketptnjiimtktbpjdqptdthmuMvgdt\
            b3a3qd  (61 chars)
  LDUDE   : uAurftmtg2q2hbrhcbbmfcepnjiimidpjdqpmrmuMuqmb3a3qd
            (51 chars, 10 chars (16%) shorter)
```

```
(L3) Czech:  (using Latin Extended A)
  Pro<ccaron>prost<ecaron>nemluv<iacute><ccaron>esky
  U+0050 u+0072 u+006F u+010D u+0070 u+0072 u+006F u+0073 u+0074
  u+011B u+006E u+0065 u+006D u+006C u+0075 u+0076 u+00ED u+010D
  u+0065 u+0073 u+006B u+0079

  DUDE-02 : vAuctptyctzpctptnhtyrtzfmibtjd3mt8atyitgtitc
            (45 chars)
  LDUDE   : uAukfycypkfepzpzfmibmtb3m8ayiqtik
            (34 chars, 24% shorter)
```

Security considerations

ACE-encoded reordered code points are restored in reverse ACE translation and this improvement do not introduce any new security problems into ACE.

References

[DUDE02] Mark Welter, Brian Spolarich, Adam Costello, "DUDE: Differential Unicode Domain Encoding", 2001-May-31, draft-ietf-idn-dude-02.

[AMCACEW] Adam Costello, "AMC-ACE-W version 0.1.0", 2001-May-31, draft-ietf-idn-amc-ace-w-00, latest version at http://www.cs.berkeley.edu/~amc/charset/amc-ace-w.

[UNICODE] The Unicode Consortium, "The Unicode Standard", http://www.unicode.org/unicode/standard/standard.html.

[IDNA] Patrik Falstrom, Paul Hoffman, "Internationalizing Host Names In Applications (IDNA)", draft-ietf-idn-idna-01

[NAMEPREP] Paul Hoffman, Marc Blanchet, "Preparation of Internationalized Host Names", Feb 2001, draft-ietf-idn-nameprep-03

Author

Soobok Lee <lsb@postel.co.kr>
Postel Services, Inc.
http://www.postel.co.kr
Tel: +82-11-9774-2737

LDUDE: Example implementation into DUDE-02

This idea is applicable to any ACEs.
LDUDE is a name for DUDE-02 implementation of this idea.

Embedded hangul,han and Latin frequency tables are subject to change with further studies in the next revision of this draft.

In Unix, save this example source code into ldude.c

```
% cc -o ldude ldude.c
% ./ldude -e < input_file > output_file
% ./ldude -d < output_file
```

An input file should contains u+????-form code points delimited with spaces or newlines.

```
/* begin of ldude.c */

/*******************************************************/
/* ldude.c 1.0  (2001-Jul-3)                           */
/* Soobok Lee <lsb@postel.co.kr>                       */
/* dude.c from Adam M. Costello <amc@cs.berkeley.edu> */
/*******************************************************/

/* This is ANSI C code (C89) implementing */
/* DUDE (draft-ietf-idn-ldude-01).        */


/*************************************************************/
/* Public interface (would normally go in its own .h file): */

#include <stdio.h>
```

```
#include <limits.h>

enum dude_status {
  dude_success,
  dude_bad_input,
  dude_big_output  /* Output would exceed the space provided. */
};

enum case_sensitivity { case_sensitive, case_insensitive };

#if UINT_MAX >= 0x1FFFFF
typedef unsigned int u_code_point;
#else
typedef unsigned long u_code_point;
#endif

enum dude_status dude_encode(
  unsigned int input_length,
  const u_code_point input[],
  const unsigned char uppercase_flags[],
  unsigned int *output_size,
  char output[] );

    /* dude_encode() converts Unicode to DUDE (without any            */
    /* signature).  The input must be represented as an array         */
    /* of Unicode code points (not code units; surrogate pairs        */
    /* are not allowed), and the output will be represented as        */
    /* null-terminated ASCII.  The input_length is the number of code */
    /* points in the input.  The output_size is an in/out argument:   */
    /* the caller must pass in the maximum number of characters       */
    /* that may be output (including the terminating null), and on    */
    /* successful return it will contain the number of characters     */
    /* actually output (including the terminating null, so it will be */
    /* one more than strlen() would return, which is why it is called */
    /* output_size rather than output_length).  The uppercase_flags   */
    /* array must hold input_length boolean values, where nonzero     */
    /* means the corresponding Unicode character should be forced     */
    /* to uppercase after being decoded, and zero means it is         */
    /* caseless or should be forced to lowercase.  Alternatively,     */
    /* uppercase_flags may be a null pointer, which is equivalent     */
    /* to all zeros.  The encoder always outputs lowercase base-32    */
    /* characters except when nonzero values of uppercase_flags       */
    /* require otherwise.  The return value may be any of the         */
    /* dude_status values defined above; if not dude_success, then    */
    /* output_size and output may contain garbage.  On success, the   */
    /* encoder will never need to write an output_size greater than   */
    /* input_length*k+1 if all the input code points are less than 1  */
    /* << (4*k), because of how the encoding is defined.              */

enum dude_status dude_decode(
  enum case_sensitivity case_sensitivity,
```

```
  char scratch_space[],
  const char input[],
  unsigned int *output_length,
  u_code_point output[],
  unsigned char uppercase_flags[] );

    /* dude_decode() converts DUDE (without any signature) to         */
    /* Unicode.  The input must be represented as null-terminated     */
    /* ASCII, and the output will be represented as an array of       */
    /* Unicode code points.  The case_sensitivity argument influences */
    /* the check on the well-formedness of the input string; it       */
    /* must be case_sensitive if case-sensitive comparisons are       */
    /* allowed on encoded strings, case_insensitive otherwise.        */
    /* The scratch_space must point to space at least as large        */
    /* as the input, which will get overwritten (this allows the      */
    /* decoder to avoid calling malloc()).  The output_length is      */
    /* an in/out argument: the caller must pass in the maximum        */
    /* number of code points that may be output, and on successful    */
    /* return it will contain the actual number of code points        */
    /* output.  The uppercase_flags array must have room for at       */
    /* least output_length values, or it may be a null pointer if     */
    /* the case information is not needed.  A nonzero flag indicates  */
    /* that the corresponding Unicode character should be forced to   */
    /* uppercase by the caller, while zero means it is caseless or    */
    /* should be forced to lowercase.  The return value may be any    */
    /* of the dude_status values defined above; if not dude_success,  */
    /* then output_length, output, and uppercase_flags may contain    */
    /* garbage.  On success, the decoder will never need to write     */
    /* an output_length greater than the length of the input (not     */
    /* counting the null terminator), because of how the encoding is  */
    /* defined.                                                       */


/*****************************************************/
/* Implementation (would normally go in its own .c file): */

#include <string.h>

/* Character utilities: */

/* base32[q] is the lowercase base-32 character representing  */
/* the number q from the range 0 to 31.  Note that we cannot  */
/* use string literals for ASCII characters because an ANSI C */
/* compiler does not necessarily use ASCII.                   */

static const char base32[] = {
  97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,     /* a-k */
  109, 110,                                               /* m-n */
  112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,  /* p-z */
  50, 51, 52, 53, 54, 55, 56, 57                          /* 2-9 */
};
```

```
/* base32_decode(c) returns the value of a base-32 character, in the */
/* range 0 to 31, or the constant base32_invalid if c is not a valid */
/* base-32 character.                                                */

enum { base32_invalid = 32 };

static unsigned int base32_decode(char c)
{
  if (c < 50) return base32_invalid;
  if (c <= 57) return c - 26;
  if (c < 97) c += 32;
  if (c < 97 || c == 108 || c == 111 || c > 122) return base32_invalid;
  return c - 97 - (c > 108) - (c > 111);
}

/* unequal(case_sensitivity,s1,s2) returns 0 if the strings s1 and s2 */
/* are equal, 1 otherwise.  If case_sensitivity is case_insensitive,  */
/* then ASCII A-Z are considered equal to a-z respectively.           */

static int unequal( enum case_sensitivity case_sensitivity,
                    const char s1[], const char s2[]        )
{
  char c1, c2;

  if (case_sensitivity != case_insensitive) return strcmp(s1,s2) != 0;

  for (;;) {
    c1 = *s1;
    c2 = *s2;
    if (c1 >= 65 && c1 <= 90) c1 += 32;
    if (c2 >= 65 && c2 <= 90) c2 += 32;
    if (c1 != c2) return 1;
    if (c1 == 0) return 0;
    ++s1, ++s2;
  }
}



/* LANGUAGE-SPECIFIC IMPROVEMENTS TO DUDE  BASED ON CODE REORDERING */

int isHANGUL(u_code_point s) {
        int SIndex = s - 0xAC00;
        if (SIndex < 0 || SIndex >= 11172) {
            return 0;
        }
        return 1;
};
int isUNIHAN(u_code_point s) {
        if (s >= 0x4E00 && s <= 0x9FAF) {
            return 1;
```

```
        }
        return 0;
};
int isKATAKANA(u_code_point s) {
        if (s >= 0x30A0 && s <= 0x30FF) {
            return 1;
        }
        return 0;
};
int isHINDI(u_code_point s) {
        if (s >= 0x0900 && s <= 0x0970) {
            return 1;
        }
        return 0;
};
int isLatins(u_code_point s) {
        if (s <  0x370) {
            return 1;
        }
        return 0;
};


// Most frequent 888 Hangeul syllables in Korean BizName
#define HG 888
u_code_point hangeul_freq[HG] = {

 0xd55c,0xad6d,0xd559,0xad50,0xb300,0xace0,0xb4f1,0xcd08,
 0xc911,0xb824,0xd654,0xd604,0xc6d0,0xbb38,0xc721,0xbcd1,
 0xc804,0xc790,0xae30,0xacf5,0xc0b0,0xc5c5,0xacc4,0xbb3c,
 0xb958,0xc6b4,0xb3d9,0xcc28,0xc220,0xd56d,0xbd80,0xd68d,
 0xac74,0xc124,0xcee8,0xd305,0xac15,0xc0dd,0xba85,0xc885,
 0xd569,0xc601,0xb18d,0xbb34,0xc5ed,0xc5f0,0xb9f9,0xc120,
 0xc11c,0xc6b8,0xbe44,0xc2dc,0xc2a4,0xd15c,0xd14d,0xd0dd,
 0xc8fc,0xc2dd,0xd3ec,0xce20,0xbc30,0xb2ec,0xc368,0xaf43,
 0xc815,0xbcf4,0xd1b5,0xc2e0,0xc0c1,0xc0ac,0xd68c,0xc138,
 0xc6a9,0xd611,0xcd9c,0xd310,0xc9c4,0xb791,0xb9e4,0xd5d8,
 0xb0b4,0xc154,0xc1fc,0xd551,0xb0a0,0xb110,0xb370,0xc774,
 0xd648,0xb9c8,0xbc14,0xc624,0xc0bf,0xc9d0,0xc2ed,0xc548,
 0xc18c,0xd504,0xd2b8,0xc6e8,0xbbf8,0xb514,0xc5b4,0xc544,
 0xd53c,0xd30c,0xcf54,0xb9ac,0xceec,0xce7c,0xcf00,0xba54,
 0xd22c,0xc740,0xd589,0xce74,0xb4dc,0xadf8,0xb8f9,0xb9b0,
 0xc6d4,0xb79c,0xc5ec,0xc88b,0xace8,0xce90,0xb9bc,0xd578,
 0xac1c,0xbc1c,0xc5d8,0xc9c0,0xae00,0xb85c,0xbc8c,0xc810,
 0xd574,0xd138,0xd0c8,0xd1a0,0xd3f0,0xc678,0xacfc,0xc694,
 0xc778,0xb137,0xb2f7,0xd154,0xb808,0xcf64,0xcef4,0xd4e8,
 0xd130,0xc5d4,0xd14c,0xbc45,0xd06c,0xc13c,0xb2e5,0xd0c0,
 0xc7a5,0xc57d,0xd488,0xc81c,0xc194,0xb8e8,0xc158,0xbc29,
 0xc1a1,0xc77c,0xd074,0xb7fd,0xb355,0xd615,0xd328,0xd3c9,
 0xc0bc,0xc131,0xb0a8,0xbd81,0xac8c,0xc784,0xd50c,0xb77c,
 0xc6cc,0xb7ec,0xc704,0xc628,0xd658,0xacbd,0xcda9,0xbdf0,
```

0xc1c4,0xc564,0xc528,0xc640,0xce58,0xb125,0xc5d0,0xc5e0,
0xd050,0xc54c,0xd2f0,0xc720,0xbe0c,0xc5d1,0xbe14,0xd29c,
0xbcc0,0xd638,0xbc95,0xb960,0xae08,0xad11,0xcc9c,0xc18d,
0xc591,0xd65c,0xccad,0xc988,0xc139,0xd734,0xcf5c,0xb354,
0xd0dc,0xd398,0xb274,0xb9e5,0xbca8,0xcd95,0xc6f0,0xbca0,
0xb860,0xb2c9,0xad7f,0xc9c1,0xc2f8,0xc820,0xbe5b,0xc758,
0xbc84,0xc6f9,0xd558,0xac00,0xc744,0xbc31,0xb124,0xd035,
0xc288,0xc218,0xd37c,0xcee4,0xbba4,0xb2c8,0xb9c1,0xb450,
0xbbfc,0xb4e0,0xb95c,0xc655,0xd45c,0xc900,0xc584,0xd2f1,
0xd765,0xd0d1,0xc870,0xbcf5,0xad6c,0xd2b9,0xbaa9,0xb78c,
0xbd09,0xd6c4,0xd0b9,0xd038,0xd48d,0xbcc4,0xc554,0xc96c,
0xd070,0xd61c,0xc5b8,0xb798,0xc560,0xbca4,0xcc98,0xd3f4,
0xaddc,0xd6fc,0xbc00,0xc5c4,0xcde8,0xb984,0xcc3d,0xc30d,
0xb2dd,0xd2f8,0xcea0,0xc824,0xc728,0xd0a4,0xc6c5,0xd64d,
0xc2e4,0xc708,0xd30d,0xcc38,0xd5e4,0xb7f4,0xc625,0xad00,
0xb3cc,0xc608,0xd380,0xc62c,0xc2b9,0xc11d,0xb839,0xb9db,
0xc4f0,0xc0e4,0xadf9,0xd5a5,0xd53d,0xb80c,0xd718,0xb9de,
0xcda4,0xbe4c,0xcd94,0xb9cc,0xd1b1,0xb108,0xafbc,0xba38,
0xc6b0,0xc724,0xd329,0xd480,0xc82f,0xc874,0xc8e4,0xce85,
0xb4e4,0xbcf8,0xbc94,0xb825,0xc559,0xaca8,0xcfe0,0xd584,
0xb3c4,0xb098,0xbaa8,0xb2e4,0xc7ac,0xad8c,0xb178,0xbab0,
0xb2e8,0xc9d1,0xccb4,0xc74c,0xb8cc,0xc99d,0xac70,0xae40,
0xb2f9,0xc57c,0xb974,0xbc15,0xc800,0xac80,0xc785,0xb529,
0xb86f,0xcca0,0xbd88,0xbc18,0xbc88,0xc775,0xbd84,0xc791,
0xc0f5,0xb9ad,0xba55,0xac04,0xad70,0xd6a8,0xb2f4,0xb204,
0xcf58,0xd478,0xc0c8,0xd560,0xac10,0xd0c1,0xcfe8,0xc5fc,
0xc5f4,0xac08,0xc545,0xd5c8,0xd544,0xb809,0xd63c,0xb294,
0xb3c5,0xd568,0xcf13,0xc0c9,0xcd0c,0xb4c0,0xb7ed,0xac01,
0xc735,0xb780,0xc2ec,0xba74,0xba3c,0xaca9,0xce68,0xc871,
0xd76c,0xd669,0xd5ec,0xcc44,0xc9c8,0xc789,0xc561,0xb0c9,
0xb840,0xc83c,0xb208,0xd314,0xcc30,0xc801,0xc555,0xacac,
0xd640,0xc8fd,0xc808,0xbe59,0xd540,0xc5bc,0xc2f1,0xb864,
0xadfc,0xd5cc,0xc300,0xc190,0xbe45,0xac1d,0xd0a8,0xcc99,
0xc2ac,0xb09a,0xad74,0xce60,0xc811,0xc2a8,0xc26c,0xb9bd,
0xb85d,0xb784,0xb179,0xace1,0xacb0,0xd2bc,0xd134,0xd0c4,
0xce5c,0xcc45,0xcc2c,0xc6cd,0xc6c0,0xc568,0xc12c,0xb77d,
0xd3b8,0xd32c,0xd150,0xc7a1,0xbe48,0xb9d0,0xb7c9,0xb180,
0xd38c,0xbbf9,0xbaac,0xba40,0xb989,0xb799,0xb144,0xae38,
0xce21,0xc6c3,0xc308,0xc12f,0xc0b4,0xbc0d,0xb978,0xb760,
0xb378,0xb09c,0xd034,0xbc25,0xb9dd,0xb728,0xb2a5,0xb290,
0xd790,0xcd98,0xc637,0xc21c,0xb9e8,0xb9d8,0xb298,0xb150,
0xae09,0xac24,0xd2c0,0xcea1,0xc20d,0xc1e0,0xbcbd,0xbc38,
0xb871,0xb81b,0xb7a8,0xb304,0xd6c8,0xd3ed,0xd0f1,0xcf10,
0xcef5,0xcd5c,0xcd1d,0xc82c,0xc36c,0xc140,0xc0d8,0xbe75,
0xbe60,0xbe10,0xbd95,0xb7f0,0xb7b5,0xb610,0xb3c8,0xb374,
0xb12c,0xb099,0xb044,0xd788,0xd2f4,0xd1a4,0xd0d0,0xc9dc,
0xc58f,0xc2b4,0xc1a5,0xb3d4,0xafc0,0xadc0,0xd508,0xd3fc,
0xd3d0,0xd39c,0xd399,0xd31c,0xd1a8,0xd131,0xce94,0xcd09,
0xccd0,0xcca8,0xcc60,0xcc3e,0xcc29,0xc9f8,0xc9d5,0xc81d,
0xc7a0,0xc644,0xc2b5,0xbc34,0xb9c9,0xb828,0xb2d8,0xb205,
0xae4c,0xd608,0xd31d,0xc90c,0xc88c,0xc73c,0xc5fd,0xc14b,

```
 0xc0f7,0xbc1d,0xba64,0xb561,0xb524,0xb118,0xb0ad,0xb07c,
 0xade0,0xac9c,0xac78,0xcfe1,0xcf69,0xcf04,0xc9f1,0xc695,
 0xc573,0xc55e,0xc53d,0xc329,0xc290,0xc19c,0xc0ad,0xbb18,
 0xb86c,0xb7fc,0xb545,0xb17c,0xaebc,0xae68,0xacf6,0xd799,
 0xd761,0xd655,0xd5db,0xd56b,0xd1f4,0xd0b4,0xce78,0xcc0c,
 0xc990,0xc63b,0xc61b,0xc384,0xbd99,0xbd90,0xbcfc,0xb8e9,
 0xb7a9,0xb69c,0xb5cc,0xb5a1,0xb518,0xb515,0xb451,0xb3fc,
 0xb371,0xb358,0xb2ed,0xb188,0xb0e5,0xaf42,0xace4,0xd720,
 0xd700,0xd234,0xd1a1,0xcf70,0xcf08,0xce04,0xc9d3,0xc98c,
 0xc813,0xc7bc,0xc70c,0xc570,0xc500,0xc3e0,0xc3d8,0xc2f9,
 0xc27d,0xc250,0xc22f,0xc058,0xbe68,0xbe54,0xbcbc,0xbabd,
 0xba58,0xba4d,0xb9b4,0xb8f8,0xb460,0xb380,0xb1cc,0xb192,
 0xb140,0xb128,0xb0c5,0xb0a9,0xb05d,0xaf2c,0xae54,0xad34,
 0xac90,0xd575,0xd401,0xd3a8,0xd1b0,0xd0e0,0xcfc4,0xccbc,
 0xcc4c,0xcc1c,0xcbd4,0xc9da,0xc989,0xc717,0xc635,0xc5ff,
 0xc232,0xbafc,0xb8b0,0xb7ad,0xb5bc,0xb530,0xb4dd,0xb465,
 0xb41c,0xb2d0,0xb057,0xb04c,0xad81,0xac13,0xd749,0xd6cc,
 0xd6a1,0xd601,0xd5f4,0xd54c,0xd47c,0xd3ab,0xd384,0xd31f,
 0xd300,0xd15d,0xd140,0xd0ed,0xd0ec,0xcffc,0xcf8c,0xce89,
 0xce84,0xce75,0xce69,0xcd78,0xcd2c,0xcc10,0xc9dd,0xc999,
 0xc8e0,0xc878,0xc7dd,0xc7c1,0xc7ad,0xc7a3,0xc794,0xc641,
 0xc639,0xc610,0xc5b5,0xc58d,0xc575,0xc530,0xc38c,0xc2f6,
 0xc2ef,0xc258,0xc22d,0xc219,0xc0cc,0xc0b6,0xbfcc,0xbf55,
 0xbe7c,0xbe57,0xbdd4,0xbd24,0xbca7,0xbc1f,0xbc1b,0xbbac,
 0xbab8,0xba67,0xb9f7,0xb9d1,0xb9bf,0xb98e,0xb987,0xb86d,
 0xb81d,0xb818,0xb801,0xb730,0xb6f0,0xb6b1,0xb54c,0xb534,
 0xb454,0xb3cb,0xb385,0xb364,0xb2f5,0xb2db,0xb214,0xb18b,
 0xb11d,0xb0c4,0xb0b5,0xaee8,0xae45,0xacfd,0xac71,0xac19,
 0xac11,0xd79d,0xd78c,0xd69f,0xd48b,0xd3a0,0xd301,0xd0e4,
 0xd0d5,0xd03c,0xcf65,0xcf1c,0xcea3,0xcd1b,0xcc64,0xcabd,
 0xc9c7,0xc950,0xc918,0xc8c4,0xc80a,0xc7c8,0xc74d,0xc719,
 0xc6b1,0xc651,0xc619,0xc5e3,0xc580,0xc557,0xc52c,0xc388,
 0xc2fc,0xc19d,0xc178,0xc174,0xc0ec,0xc0d0,0xc068,0xbf08,
 0xbed0,0xbcd5,0xbc40,0xbc2d,0xbbff,0xbbc0,0xbb58,0xbb44,
 0xba5c,0xba4b,0xba39,0xb9f5,0xb9d9,0xb97c,0xb959,0xb93c,
 0xb8e1,0xb819,0xb738,0xb527,0xb51c,0xb458,0xb284,0xb1e8
};


#define HANGUL_REORDER_BASE 0XB000

u_code_point reorder_hangul(u_code_point s) {
        u_code_point i=HANGUL_REORDER_BASE;
        int k=0;
        for(k=0; k<HG; k++,i++) {
           if(s == hangeul_freq[k]) { return i; };
        };
        k=(s - HANGUL_REORDER_BASE);
        if( k>=0 && k<HG) {
           return hangeul_freq[k];
        };
```

```
        return s;
}

u_code_point restore_order_hangul(u_code_point z) {
        u_code_point i=HANGUL_REORDER_BASE;
        int k;
        k=(z - HANGUL_REORDER_BASE);
        if( k>=0 && k<HG) {
           return hangeul_freq[k];
        };
        for(k=0; k<HG; k++,i++) {
           if(z == hangeul_freq[k]) { return i; };
        };
        return z;
}

//Most frequent 4096 SC/TC characters in CJK
#define UH 4096
u_code_point unihan_freq[UH] = {

 0x4f01,0x696d,0x4e1a,0x5de5,0x7a0b,0x96c6,0x5718,0x56e2,
 0x6709,0x9650,0x8cac,0x8d23,0x4efb,0x516c,0x53f8,0x603b,
 0x90e8,0x767c,0x53d1,0x5c55,0x7ad9,0x70b9,0x958b,0x5f00,
 0x79d1,0x6280,0x8853,0x672f,0x54a8,0x8be2,0x5be6,0x5b9e,
 0x901a,0x4fe1,0x606f,0x7cfb,0x7edf,0x7d71,0x7db2,0x8def,
 0x7edc,0x4e2d,0x5fc3,0x7f51,0x56fd,0x570b,0x969b,0x83ef,
 0x96fb,0x7535,0x5b50,0x8111,0x8166,0x6c23,0x6c14,0x5668,
 0x6a5f,0x6c17,0x529b,0x673a,0x68b0,0x8baf,0x8d44,0x8a71,
 0x8bbe,0x8ba1,0x8a2d,0x8a08,0x5099,0x5408,0x52d5,0x52a8,
 0x5236,0x88fd,0x9020,0x4f5c,0x5907,0x8fd0,0x5efa,0x65b0,
 0x7522,0x4ea7,0x7528,0x54c1,0x5382,0x5e94,0x793c,0x98df,
 0x79df,0x8d41,0x5ee0,0x79ae,0x5168,0x7403,0x5c08,0x7523,
 0x773c,0x955c,0x7f8e,0x5bb9,0x6c7d,0x8f66,0x8eca,0x975e,
 0x4ea4,0x6362,0x5bf9,0x5916,0x85cf,0x4e91,0x9655,0x8a9e,
 0x57df,0x540d,0x6ce8,0x518c,0x5e7f,0x64ad,0x5ee3,0x544a,
 0x4e3b,0x6e90,0x50b3,0x4e92,0x806f,0x8054,0x5149,0x6750,
 0x5927,0x5b66,0x5b78,0x5206,0x682a,0x5f0f,0x76df,0x534f,
 0x59d4,0x5458,0x4f1a,0x6703,0x793e,0x7559,0x5354,0x6559,
 0x519c,0x4e13,0x51fa,0x7248,0x6587,0x5316,0x827a,0x85dd,
 0x4f53,0x6210,0x4eba,0x624d,0x65e5,0x672c,0x9577,0x6708,
 0x751f,0x6cd5,0x5f8b,0x5e08,0x5e2b,0x8303,0x533b,0x7597,
 0x6cbb,0x836f,0x4fdd,0x5065,0x5eb7,0x8eab,0x967a,0x96aa,
 0x8d38,0x6613,0x80a1,0x4efd,0x8f6f,0x4ef6,0x8edf,0x9ad4,
 0x5a92,0x8cfc,0x7269,0x6d41,0x65c5,0x904a,0x97f3,0x6a02,
 0x670d,0x52d9,0x52a1,0x5546,0x4e8b,0x7814,0x7a76,0x6240,
 0x9662,0x88dc,0x7fd2,0x73ed,0x5100,0x60c5,0x5831,0x9023,
 0x987e,0x95ee,0x514d,0x8d39,0x53f0,0x6e7e,0x8ba4,0x8bc1,
 0x8c0d,0x7e3d,0x5834,0x573a,0x8fb2,0x89c6,0x8208,0x8cbb,
 0x91d1,0x5c5e,0x5c6c,0x92fc,0x6a21,0x9435,0x7cbe,0x5bc6,
 0x66f8,0x5c4b,0x5167,0x5e97,0x878d,0x904b,0x8f38,0x5ba2,
 0x8b49,0x5238,0x6295,0x8a17,0x9867,0x554f,0x7d9c,0x8ca1,
```

0x7d93,0x7ecf,0x7968,0x9280,0x884c,0x92b7,0x7ba1,0x7406,
0x8cb8,0x6b3e,0x5c0f,0x57fa,0x81ea,0x8aee,0x8a62,0x5275,
0x5bb6,0x5177,0x767e,0x8ca8,0x74b0,0x5883,0x76d1,0x5229,
0x7dda,0x5370,0x5237,0x5730,0x651d,0x5f71,0x88dd,0x98fe,
0x5283,0x805e,0x7b97,0x547d,0x73e0,0x5bf6,0x9418,0x9336,
0x5357,0x5dde,0x5c71,0x4e1c,0x6cb3,0x6c5f,0x6e56,0x7701,
0x5317,0x897f,0x4eac,0x5ddd,0x4e0a,0x6d77,0x4e34,0x5e02,
0x5929,0x6d25,0x8fde,0x6df1,0x5733,0x7586,0x6e29,0x6d59,
0x82cf,0x7518,0x8083,0x5b89,0x5fbd,0x590f,0x4e0b,0x6728,
0x6237,0x4f11,0x95f2,0x5b81,0x5e73,0x4e09,0x6b66,0x6c38,
0x7389,0x9633,0x8fbd,0x8d35,0x56db,0x53bf,0x5409,0x77f3,
0x592a,0x677e,0x6c99,0x66f2,0x9752,0x6e05,0x9686,0x9646,
0x4e50,0x83b1,0x666f,0x664b,0x9ec4,0x6dee,0x9e64,0x56fa,
0x9ad8,0x6607,0x961c,0x51e4,0x5b9a,0x5fb7,0x4e39,0x957f,
0x660c,0x535a,0x767d,0x963f,0x53e4,0x547c,0x60e0,0x83f1,
0x77e2,0x5d0e,0x6a2a,0x7acb,0x661f,0x6804,0x6238,0x6771,
0x6751,0x6ca2,0x80b2,0x95a2,0x7e4a,0x7dad,0x9060,0x85e4,
0x65ed,0x785d,0x68ee,0x4eca,0x6d0b,0x771f,0x5b9f,0x6e9d,
0x6b21,0x5d8b,0x798f,0x5ca1,0x5bae,0x8a0a,0x8cc7,0x7530,
0x6fa4,0x5bcc,0x58eb,0x6797,0x76f8,0x5171,0x5cf6,0x6e21,
0x702c,0x842c,0x4e16,0x6851,0x6597,0x6a4b,0x7532,0x6d5c,
0x718a,0x623f,0x7b2c,0x79cb,0x8218,0x4e80,0x962a,0x52dd,
0x7247,0x8cc0,0x524d,0x8c4a,0x6803,0x90a6,0x967d,0x6975,
0x4f50,0x5e78,0x5f18,0x8fd1,0x5f8c,0x9234,0x6749,0x7af9,
0x7279,0x6b8a,0x6839,0x88b4,0x8d8a,0x4e38,0x4f4f,0x7d00,
0x5c3e,0x8352,0x9f8d,0x6817,0x592e,0x5e83,0x5fa1,0x7a4d,
0x53cb,0x4ef2,0x80fd,0x5b87,0x83ca,0x5036,0x697d,0x68a8,
0x611b,0x77e5,0x5a9b,0x5948,0x5c90,0x7fa4,0x99ac,0x57fc,
0x9759,0x5343,0x8449,0x9ce5,0x53d6,0x826f,0x6f5f,0x5f62,
0x58f2,0x7d50,0x969c,0x5bb3,0x5199,0x4e57,0x7dcf,0x753b,
0x9580,0x6c96,0x7e04,0x757f,0x9678,0x533a,0x69cb,0x6a29,
0x52b4,0x691c,0x8b72,0x570f,0x5b85,0x8a3c,0x8cc3,0x4fa1,
0x4f9b,0x7d66,0x6bce,0x8b1b,0x6f14,0x9451,0x9031,0x520a,
0x5175,0x5eab,0x5e9c,0x770c,0x75c5,0x8a8d,0x653f,0x515a,
0x73fe,0x7d4c,0x6e08,0x9053,0x7d44,0x52a0,0x56e3,0x8ee2,
0x9f99,0x6cf0,0x4e4c,0x5434,0x5174,0x4f0a,0x5b9c,0x5cb3,
0x5f20,0x6cd7,0x91cd,0x5e86,0x9675,0x7965,0x78f4,0x76f1,
0x7719,0x53e3,0x57ce,0x8363,0x6625,0x6c60,0x6d2a,0x660e,
0x6eaa,0x5d03,0x6743,0x4e49,0x8d21,0x6e2f,0x6d66,0x6811,
0x5e84,0x5f3a,0x9704,0x548c,0x6d6e,0x6c0f,0x73af,0x59da,
0x8c0a,0x9756,0x5609,0x6d4e,0x6f6d,0x9a6c,0x95e8,0x90fd,
0x5bbe,0x5f81,0x539f,0x8c37,0x6cc9,0x5a01,0x95fb,0x6c34,
0x4f59,0x4e61,0x91ce,0x6c82,0x90d1,0x7edb,0x611f,0x6c11,
0x6843,0x5c45,0x6e38,0x5ce1,0x94a2,0x83b2,0x534e,0x965f,
0x9091,0x7a74,0x8fdb,0x6c49,0x5fe0,0x6865,0x68e3,0x5cad,
0x8f89,0x574a,0x8fdc,0x594e,0x82cd,0x8f7d,0x5e90,0x67f1,
0x6f58,0x6ecb,0x8305,0x90a1,0x5830,0x676d,0x953a,0x7a37,
0x9976,0x865e,0x9634,0x8fbe,0x768b,0x5bff,0x6000,0x82d1,
0x971e,0x679c,0x5ea6,0x51c9,0x9065,0x9526,0x666e,0x6ce2,
0x96c4,0x76ae,0x5145,0x4faf,0x4e30,0x5be8,0x5e95,0x5ca9,
0x4e95,0x74a7,0x6cad,0x7317,0x6cfd,0x9896,0x6c7e,0x8821,

0x829d,0x829c,0x9c81,0x5c01,0x8d24,0x5854,0x575b,0x5802,
0x6cb9,0x74ef,0x5cea,0x58a8,0x9a85,0x6885,0x5188,0x4ec1,
0x57a3,0x58c1,0x80a5,0x95f4,0x90f8,0x4f26,0x62c9,0x5c14,
0x59cb,0x853a,0x4e08,0x6d6a,0x6df3,0x6986,0x5510,0x7b60,
0x8981,0x90ae,0x88d5,0x987a,0x9f0e,0x6c9f,0x51f0,0x79ba,
0x65bd,0x6566,0x714c,0x5300,0x8425,0x839e,0x5934,0x80dc,
0x8fb9,0x5f92,0x8354,0x719f,0x7f57,0x9e21,0x4ead,0x57e0,
0x94f6,0x5f66,0x5df4,0x6556,0x56fe,0x52d2,0x575d,0x978d,
0x9738,0x67cf,0x868c,0x5305,0x5b9d,0x6ee8,0x52c3,0x6cca,
0x66f9,0x8336,0x5e38,0x671d,0x6f6e,0x5de2,0x90f4,0x6f84,
0x627f,0x8d64,0x5d07,0x6ec1,0x695a,0x6148,0x4ece,0x5355,
0x5f53,0x7a3b,0x767b,0x9093,0x8fea,0x6d1e,0x5ce8,0x5d4b,
0x5a25,0x9102,0x6069,0x756a,0x65b9,0x9632,0x5949,0x4f5b,
0x6276,0x629a,0x683c,0x4e2a,0x5de9,0x6842,0x54c8,0x90af,
0x542b,0x8377,0x83cf,0x8d3a,0x9ed1,0x5b88,0x8861,0x7ea2,
0x846b,0x82a6,0x864e,0x82b1,0x6ed1,0x69d0,0x83b7,0x970d,
0x7ee9,0x5373,0x5180,0x5939,0x4f73,0x7b80,0x5251,0x59dc,
0x5c06,0x7126,0x80f6,0x63ed,0x4ecb,0x8346,0x4e5d,0x9152,
0x53e5,0x5580,0x51ef,0x514b,0x57a6,0x5e93,0x6606,0x5170,
0x5eca,0x8001,0x96f7,0x51b7,0x4e3d,0x5ec9,0x6d9f,0x6881,
0x804a,0x7075,0x67f3,0x516d,0x5a04,0x9e7f,0x6f5e,0x6ee6,
0x6d1b,0x6ee1,0x8302,0x7709,0x8499,0x5b5f,0x7c73,0x7ef5,
0x95fd,0x7261,0x7a46,0x5ae9,0x76d8,0x84ec,0x5f6d,0x6c9b,
0x78d0,0x840d,0x8386,0x84b2,0x6816,0x4e03,0x9f50,0x7941,
0x542f,0x8fc1,0x6f5c,0x94a6,0x743c,0x90b1,0x5982,0x4e73,
0x6c5d,0x745e,0x838e,0x8272,0x6c55,0x5c1a,0x91b4,0x9edf,
0x97f6,0x5173,0x90b5,0x7ecd,0x5c04,0x5341,0x4ec0,0x8212,
0x53cc,0x6714,0x601d,0x5bbf,0x968f,0x7ee5,0x9042,0x68e0,
0x94c1,0x6850,0x540c,0x94dc,0x4e07,0x6c6a,0x65fa,0x671b,
0x5fae,0x6f4d,0x6e2d,0x536b,0x74ee,0x6da1,0x65e0,0x68a7,
0x4e94,0x821e,0x9521,0x53a6,0x4ed9,0x54b8,0x732e,0x9999,
0x8944,0x6e58,0x54cd,0x9879,0x8c61,0x8427,0x5b5d,0x8f9b,
0x5ffb,0x90a2,0x5f90,0x4fee,0x53d9,0x8bb8,0x859b,0x65ec,
0x5bfb,0x96c5,0x70df,0x76d0,0x5ef6,0x6cbf,0x626c,0x4eea,
0x76ca,0x82f1,0x9e70,0x79b9,0x5143,0x8d5e,0x67a3,0x589e,
0x624e,0x5c6f,0x6cbe,0x6e5b,0x6a1f,0x7ae0,0x6f33,0x62db,
0x662d,0x8d75,0x8087,0x9547,0x6b63,0x679d,0x821f,0x5468,
0x8bf8,0x9a7b,0x6dc4,0x7d2b,0x90b9,0x9075,0x5de6,0x5043,
0x510b,0x5156,0x4eb3,0x9097,0x90b3,0x90d3,0x90eb,0x90ef,
0x5152,0x7ae5,0x8297,0x82ae,0x8392,0x834f,0x8365,0x8398,
0x8572,0x5c91,0x5c9a,0x5d4a,0x5d69,0x8862,0x9606,0x6c76,
0x6cf8,0x6cfe,0x6d4f,0x6d60,0x6dc7,0x6dc5,0x6dbf,0x6e11,
0x6e5f,0x6e44,0x6ea7,0x6f62,0x6fa7,0x6fee,0x7f19,0x9095,
0x73f2,0x679e,0x67d8,0x6866,0x683e,0x6ed5,0x65cc,0x7800,
0x7684,0x662f,0x4e00,0x4e0d,0x6211,0x4e86,0x5728,0x5230,
0x4ed6,0x4f60,0x4ee5,0x53ef,0x5c31,0x4e5f,0x597d,0x8fd9,
0x90a3,0x5f97,0x0000,0x6765,0x4e4b,0x5e74,0x53bb,0x591a,
0x770b,0x9019,0x500b,0x800c,0x60f3,0x8bf4,0x4eec,0x70ba,
0x53ea,0x4f86,0x7136,0x4e3a,0x63d0,0x5979,0x65f6,0x6642,
0x4f46,0x5f88,0x8aaa,0x6c92,0x8d77,0x624b,0x610f,0x53c8,
0x4e9b,0x904e,0x5176,0x9762,0x8acb,0x7740,0x5011,0x6b64,

0x6700,0x8fc7,0x91cc,0x5df2,0x4f55,0x56e0,0x9ebc,0x8005,
0x4e8c,0x540e,0x4f4d,0x9084,0x5c0d,0x5973,0x4e48,0x5df1,
0x56de,0x628a,0x518d,0x6253,0x6bd4,0x6ca1,0x4f7f,0x4e8e,
0x88ab,0x7b49,0x8fd8,0x5c11,0x6216,0x7121,0x65bc,0x6027,
0x5427,0x7576,0x5411,0x55ce,0x5148,0x5404,0x7531,0x5165,
0x89c1,0x53ca,0x4fbf,0x505a,0x50cf,0x671f,0x4ee3,0x76ee,
0x89e3,0x9ede,0x984c,0x8868,0x5462,0x8d70,0x4e24,0x66f4,
0x6a23,0x81f3,0x6837,0x73b0,0x5b83,0x6d3b,0x4e0e,0x600e,
0x795e,0x653e,0x6821,0x8b1d,0x8457,0x5feb,0x63a5,0x6b7b,
0x53cd,0x8207,0x738b,0x5b57,0x53d7,0x79cd,0x58f0,0x7b11,
0x627e,0x76f4,0x53eb,0x8bdd,0x513f,0x6bcf,0x8a00,0x61c9,
0x7a2e,0x5b8c,0x6307,0x51e0,0x7ed9,0x529f,0x559c,0x82e5,
0x5f1f,0x8ddf,0x95dc,0x754c,0x60a8,0x9593,0x9cf3,0x5fc5,
0x89ba,0x8a72,0x6539,0x5426,0x516b,0x7a7a,0x554a,0x9032,
0x5566,0x5403,0x4e14,0x5c07,0x5169,0x7537,0x8003,0x6c42,
0x542c,0x5e76,0x8ad6,0x5185,0x672a,0x5225,0x807d,0x6301,
0x5019,0x898b,0x88e1,0x98a8,0x5374,0x519b,0x7063,0x91cf,
0x534a,0x5e0c,0x5f80,0x522b,0x5904,0x674e,0x73a9,0x66fe,
0x5931,0x5340,0x4e66,0x932f,0x5b69,0x54ea,0x6536,0x8b93,
0x62ff,0x4ee4,0x9078,0x62a5,0x8f03,0x751a,0x6578,0x652f,
0x5f9e,0x4f3c,0x6b61,0x96be,0x6570,0x6bdb,0x6b65,0x65e9,
0x822c,0x5e7e,0x706b,0x9700,0x53e6,0x592b,0x4e4e,0x96e3,
0x982d,0x5ba4,0x6599,0x5012,0x8a31,0x4eb2,0x6574,0x5e72,
0x8cb7,0x8a18,0x5144,0x865f,0x670b,0x843d,0x8655,0x9996,
0x65af,0x9664,0x6bb5,0x6015,0x5ff5,0x6545,0x793a,0x63a8,
0x4e45,0x5947,0x4e26,0x7236,0x5f35,0x665a,0x5207,0x8bb0,
0x7834,0x53f2,0x5fd7,0x8ab0,0x98ce,0x7167,0x6218,0x7adf,
0x5f15,0x54e5,0x89c9,0x9898,0x5f85,0x6848,0x8bf7,0x5b58,
0x7231,0x8ba9,0x5c40,0x591c,0x82e6,0x7b54,0x901f,0x6b4c,
0x9673,0x8bba,0x8f49,0x9ee8,0x6d3e,0x5361,0x8b8a,0x8a66,
0x6d88,0x7ed3,0x602a,0x8db3,0x677f,0x5dee,0x55ae,0x7fa9,
0x5217,0x578b,0x9769,0x6230,0x961f,0x5750,0x968a,0x537b,
0x6392,0x5e26,0x8d85,0x5047,0x9001,0x5beb,0x5b98,0x6761,
0x8072,0x53d8,0x8be5,0x81fa,0x9886,0x4f20,0x6bcd,0x54e1,
0x6389,0x8a0e,0x67e5,0x5247,0x51b3,0x6a94,0x5475,0x4f4e,
0x4ecd,0x59b3,0x529e,0x521d,0x5e03,0x5f37,0x8b70,0x52a9,
0x8fa6,0x50f9,0x571f,0x8f6c,0x505c,0x4f17,0x8f7b,0x5ea7,
0x503c,0x6562,0x8bed,0x65cf,0x8ff7,0x7a81,0x53f3,0x6c7a,
0x67d0,0x8bc6,0x6781,0x7d1a,0x8840,0x8036,0x820d,0x8138,
0x8dd1,0x94b1,0x523b,0x6025,0x4f9d,0x5594,0x6551,0x6a19,
0x7368,0x5386,0x89d2,0x5fd8,0x8c93,0x6548,0x75db,0x9ec3,
0x53c3,0x4f8b,0x8bae,0x8996,0x89c0,0x51c6,0x8863,0x9645,
0x5219,0x6279,0x636e,0x6162,0x5bfc,0x638c,0x9322,0x5531,
0x5fd9,0x80cc,0x6982,0x5473,0x5200,0x7591,0x9304,0x8bfb,
0x98de,0x89c2,0x4e89,0x5e1d,0x63db,0x7ec4,0x81f4,0x6309,
0x79bb,0x867d,0x6b62,0x786c,0x7f16,0x5e6b,0x78ba,0x8c08,
0x8ffd,0x7387,0x5c3d,0x8bb2,0x985e,0x6740,0x756b,0x8c03,
0x8a34,0x9047,0x6fc0,0x559d,0x65e2,0x5e36,0x667a,0x9644,
0x6697,0x7ec8,0x65c1,0x80e1,0x59b9,0x59d0,0x8da3,0x7ea7,
0x5716,0x68d2,0x7bc7,0x8cfd,0x7761,0x8b58,0x908a,0x914d,
0x6bd2,0x96e8,0x51b2,0x96d6,0x4eae,0x6b0a,0x5584,0x9a57,

0x4e3e,0x6293,0x5a18,0x8349,0x8b80,0x8df3,0x98db,0x561b,
0x5440,0x70ed,0x6eff,0x5922,0x5ba3,0x8ab2,0x8ecd,0x79f0,
0x7f6a,0x7d04,0x7a7f,0x7ea6,0x9858,0x60ca,0x5417,0x9000,
0x653b,0x9054,0x53f7,0x90ed,0x7edd,0x9009,0x7d20,0x53c2,
0x8b66,0x4e9a,0x590d,0x4f24,0x7c7b,0x5e2d,0x5bc4,0x6b22,
0x725b,0x52bf,0x65ad,0x9648,0x61c2,0x5920,0x5348,0x4ef7,
0x5224,0x789f,0x59d3,0x62b1,0x8ac7,0x8ce3,0x89c4,0x5988,
0x521a,0x663e,0x5b97,0x6e96,0x6c89,0x5747,0x8089,0x613f,
0x6cc1,0x786e,0x724c,0x96e2,0x6388,0x4ea6,0x5c0e,0x72d7,
0x7d27,0x5e2e,0x4f2f,0x7ebf,0x9760,0x5a5a,0x8abf,0x526f,
0x6768,0x8857,0x50b7,0x525b,0x541b,0x8282,0x83ab,0x5957,
0x5509,0x88c5,0x7f6e,0x54b1,0x6bba,0x5ffd,0x5c81,0x6563,
0x7b56,0x689d,0x60b2,0x4e25,0x72c2,0x7d55,0x62dc,0x7a31,
0x7eaa,0x64da,0x811a,0x76e1,0x9732,0x6807,0x70c8,0x5712,
0x5c3c,0x996d,0x6050,0x641e,0x59d1,0x72af,0x5bdf,0x8ff0,
0x96d9,0x63a7,0x51b5,0x7d05,0x6b32,0x51fb,0x55ef,0x4ec5,
0x7a97,0x5a46,0x5347,0x6838,0x77ed,0x7eed,0x7687,0x57f7,
0x7565,0x72ec,0x66b4,0x67b6,0x4e70,0x62a4,0x9b54,0x96f2,
0x7aef,0x7f3a,0x91c7,0x7956,0x9808,0x5fcd,0x6d32,0x9b3c,
0x8cea,0x80af,0x8077,0x4e71,0x62cd,0x5fa9,0x96ea,0x5218,
0x7bc0,0x898f,0x7562,0x5f04,0x71b1,0x9ebb,0x9928,0x7237,
0x5212,0x6297,0x614b,0x4ed8,0x552e,0x89aa,0x4e82,0x5f69,
0x62ec,0x5634,0x5178,0x9519,0x521b,0x64ca,0x8209,0x987b,
0x7d42,0x7533,0x79fb,0x8239,0x6458,0x65b7,0x8f15,0x7c21,
0x97ff,0x96a8,0x7df4,0x5e55,0x7e8c,0x9b5a,0x54ed,0x804c,
0x7ec6,0x8bc9,0x6001,0x79c1,0x964d,0x7b14,0x656c,0x5757,
0x77a7,0x79c0,0x60dc,0x5e79,0x9910,0x5c0a,0x5de8,0x8d28,
0x7f85,0x7981,0x9ed8,0x5438,0x907f,0x97e6,0x56f0,0x56f4,
0x83dc,0x5446,0x56ed,0x6731,0x6a13,0x54c7,0x501f,0x7169,
0x591f,0x8d5b,0x6f2b,0x4fca,0x986f,0x8f83,0x78bc,0x9192,
0x675f,0x5c24,0x697c,0x5b59,0x7238,0x7d22,0x523a,0x5077,
0x552f,0x8bd7,0x8056,0x5cf0,0x58de,0x704c,0x654c,0x8bd5,
0x9810,0x8c22,0x51e1,0x773e,0x504f,0x4f38,0x722d,0x9a8c,
0x8aa4,0x6df7,0x5ead,0x5806,0x9806,0x8033,0x9aa8,0x517b,
0x8cb4,0x900f,0x8ca0,0x58d3,0x6076,0x9069,0x4eab,0x4fc2,
0x7ef4,0x51b0,0x6e2c,0x6e10,0x61f7,0x5de7,0x8fce,0x5360,
0x79d8,0x5f02,0x6d17,0x55da,0x8d1f,0x4ea1,0x8a55,0x9635,
0x5c42,0x7d30,0x5e8f,0x9003,0x5b63,0x4f19,0x91ab,0x7ec7,
0x9986,0x904d,0x5e8a,0x7434,0x4e60,0x775b,0x7763,0x6200,
0x5f52,0x4e01,0x63f4,0x67d4,0x6557,0x4e1d,0x5371,0x7a3f,
0x694a,0x5740,0x51a0,0x723d,0x6b23,0x62bd,0x52b3,0x684c,
0x59bb,0x5987,0x6298,0x9748,0x52c7,0x6068,0x9a0e,0x4ed4,
0x8bc4,0x9014,0x9805,0x6232,0x63a2,0x5565,0x7686,0x5fb5,
0x6311,0x6beb,0x8c6a,0x52aa,0x672b,0x6258,0x53f6,0x72d0,
0x86cb,0x6628,0x538b,0x71df,0x594f,0x66ff,0x5956,0x8d76,
0x6b77,0x723e,0x5f55,0x7de8,0x9707,0x5954,0x8b77,0x9f13,
0x987f,0x64cd,0x5b64,0x64c7,0x4ebf,0x6167,0x7ee7,0x7d2f,
0x6b72,0x654f,0x4f34,0x805a,0x96bb,0x4f18,0x9669,0x9818,
0x9636,0x62c5,0x63d2,0x5c0b,0x949f,0x8bbf,0x5377,0x6eab,
0x990a,0x8ba8,0x5bd2,0x6447,0x5999,0x6784,0x7ec3,0x5f31,
0x8b02,0x7570,0x906d,0x512a,0x8feb,0x6325,0x721b,0x78b0,

0x5fcc,0x63e1,0x5976,0x9694,0x60e1,0x7eb8,0x6108,0x9876,
0x72c0,0x4e58,0x5439,0x5356,0x6478,0x5433,0x795d,0x68a6,
0x8a5e,0x5287,0x96f6,0x5267,0x563f,0x817f,0x90ce,0x975c,
0x575a,0x6f02,0x5e7b,0x731c,0x73cd,0x4e9e,0x7259,0x6742,
0x5cb8,0x9010,0x9663,0x65e7,0x56b4,0x5076,0x58d8,0x4e43,
0x539a,0x52e2,0x80f8,0x79ef,0x7239,0x76db,0x7f62,0x9022,
0x862d,0x7de3,0x7c3d,0x4e88,0x558a,0x822a,0x8131,0x5f39,
0x563b,0x7ffb,0x574f,0x883b,0x5f7c,0x9c9c,0x5708,0x6bd5,
0x6234,0x5192,0x7d61,0x6469,0x54f2,0x8f2f,0x4e7e,0x65d7,
0x6b27,0x8d99,0x6790,0x5c9b,0x820a,0x68cb,0x96dc,0x8d25,
0x67aa,0x9002,0x9e97,0x865a,0x9884,0x7bb1,0x7eb7,0x9500,
0x78c1,0x9c7c,0x7206,0x7c4d,0x8173,0x528d,0x5b8b,0x6b49,
0x6241,0x5b8f,0x706f,0x72b6,0x616e,0x7d39,0x5289,0x888b,
0x8ba2,0x61b6,0x8af8,0x7b26,0x9a82,0x8f93,0x632f,0x731b,
0x8bcd,0x53ec,0x7f75,0x7d14,0x6cea,0x4fd7,0x8aa0,0x8d22,
0x7e7c,0x54e6,0x6620,0x7cca,0x585e,0x91cb,0x8ddd,0x51ac,
0x7a0d,0x74f6,0x649e,0x84c9,0x9375,0x8d95,0x780d,0x7b46,
0x8a3b,0x5269,0x71d5,0x6028,0x7f8a,0x6a39,0x500d,0x69ae,
0x5ba1,0x5899,0x5723,0x8dc3,0x966a,0x6b78,0x6267,0x5bc2,
0x6653,0x5f48,0x57f9,0x885b,0x7070,0x4e56,0x9298,0x72fc,
0x8f88,0x584a,0x5c16,0x95ea,0x9690,0x52b2,0x6f22,0x95f9,
0x5385,0x67d3,0x8521,0x8cde,0x8f09,0x6101,0x7eff,0x62d6,
0x5766,0x4f0d,0x6c88,0x6094,0x82b3,0x6155,0x989d,0x56c9,
0x8bef,0x87a2,0x8010,0x5049,0x85a6,0x7eb3,0x8c46,0x6c61,
0x555f,0x80a9,0x62b5,0x9057,0x71c8,0x6aa2,0x65e6,0x5abd,
0x633a,0x8d27,0x8e0f,0x50bb,0x62d4,0x4ec7,0x7f13,0x8c6c,
0x4ef0,0x4f1f,0x6fdf,0x8881,0x642d,0x8a13,0x70e7,0x85cd,
0x7b28,0x6668,0x8170,0x80d6,0x62a2,0x64d4,0x88c1,0x7d19,
0x8cbc,0x620f,0x8fc5,0x6ce1,0x642c,0x8c13,0x7f77,0x8bfe,
0x8a73,0x517c,0x5b54,0x6084,0x963b,0x53d4,0x81c2,0x903c,
0x9b42,0x62e5,0x81c9,0x788e,0x53f9,0x63cf,0x4f69,0x7e41,
0x62d2,0x6302,0x54c0,0x734e,0x6ce5,0x70ae,0x7b7e,0x6575,
0x9109,0x518a,0x8f2a,0x62ac,0x8f6e,0x8bad,0x5706,0x5c3a,
0x885d,0x622a,0x91ca,0x593a,0x9df9,0x6d4b,0x76d6,0x68c4,
0x9605,0x8d2d,0x78e8,0x8000,0x5e45,0x9189,0x7e23,0x7dca,
0x7eb5,0x62e9,0x8c8c,0x50c5,0x5e33,0x5c64,0x9f20,0x9677,
0x93e1,0x5435,0x6089,0x4fc3,0x62fc,0x54e9,0x8a89,0x8d0a,
0x8986,0x978b,0x68c0,0x5bab,0x6c57,0x59ca,0x7897,0x8eb2,
0x9846,0x65cb,0x5410,0x5e7d,0x74dc,0x6de1,0x4fb5,0x9f3b,
0x8a69,0x66c9,0x6446,0x60d1,0x5965,0x6d89,0x5e3d,0x4eff,
0x64c1,0x706d,0x6176,0x7e3e,0x660f,0x8651,0x7345,0x5bbd,
0x570d,0x9918,0x5761,0x50e7,0x9b25,0x8865,0x70b8,0x7bc4,
0x8f1d,0x8b6f,0x5c41,0x72e0,0x6212,0x5ef3,0x7a33,0x722c,
0x8896,0x6c47,0x84cb,0x5211,0x7c97,0x5389,0x5c4a,0x516e,
0x8584,0x63ee,0x8ff9,0x6770,0x76fe,0x9178,0x6735,0x606d,
0x9a5a,0x78a9,0x8dcc,0x7c43,0x4e1f,0x76e4,0x6b3a,0x4e4f,
0x6355,0x6070,0x5fc6,0x54a7,0x5bfa,0x5e25,0x9080,0x8bda,
0x51cc,0x51cf,0x7384,0x865b,0x907a,0x4f0f,0x639b,0x9ebd,
0x9488,0x7ade,0x6717,0x9177,0x7c89,0x6ec5,0x609f,0x809a,
0x6691,0x8bfa,0x6b8b,0x8a8c,0x5713,0x54ac,0x5272,0x707e,
0x90aa,0x77db,0x98ef,0x4e54,0x75be,0x5a03,0x5e7c,0x7cae,

0x9802,0x8bd1,0x4fe0,0x8c0b,0x7840,0x4fc4,0x635f,0x96de,
0x8f86,0x501a,0x51c0,0x8afe,0x8f14,0x5f79,0x76c8,0x675c,
0x7bad,0x81e8,0x7f72,0x4f30,0x6170,0x80de,0x5538,0x63aa,
0x6190,0x8607,0x6e1b,0x81ed,0x51dd,0x8361,0x76fc,0x760b,
0x88c2,0x6643,0x83f2,0x594b,0x82ac,0x80c6,0x5f03,0x70e6,
0x63a1,0x5eb8,0x5c46,0x72b9,0x7a0e,0x8f29,0x85e5,0x9a19,
0x7da0,0x7e2e,0x7372,0x950b,0x62c6,0x6696,0x586b,0x50b2,
0x7262,0x60ef,0x6492,0x59c6,0x51ed,0x5e01,0x52e4,0x59a8,
0x6f38,0x659c,0x7801,0x8106,0x5ee2,0x6dda,0x6a1e,0x626f,
0x4e32,0x7a77,0x9887,0x8d0f,0x6b50,0x503e,0x5306,0x8a02,
0x97e9,0x7720,0x5587,0x98d8,0x8fb1,0x6263,0x89f8,0x8ce2,
0x79e6,0x5091,0x4fa7,0x812b,0x86c7,0x8d4f,0x7cdf,0x845b,
0x4f48,0x690d,0x6062,0x7eaf,0x95ed,0x8eba,0x62b9,0x60a0,
0x5141,0x626b,0x74e6,0x81e3,0x541f,0x84bc,0x53ad,0x6ef4,
0x5df7,0x5805,0x8d34,0x78a7,0x64e6,0x6377,0x61f6,0x6eda,
0x8e22,0x7f18,0x751c,0x8d1d,0x6da6,0x6251,0x8fd4,0x6905,
0x6d69,0x7a69,0x6269,0x73b2,0x680f,0x7272,0x5413,0x4fe9,
0x84dd,0x7d72,0x5875,0x6163,0x6fe4,0x8abc,0x4f54,0x9a91,
0x6350,0x5c60,0x626d,0x8cf4,0x968e,0x62fe,0x8c50,0x989c,
0x8d2b,0x8c9d,0x5018,0x90f5,0x85c9,0x6cdb,0x58ee,0x8428,
0x4ff1,0x5978,0x96b1,0x75bc,0x5fe7,0x9ece,0x8150,0x6158,
0x6454,0x7f9e,0x832b,0x9b4f,0x7d0d,0x827e,0x5bde,0x72f1,
0x89e6,0x6930,0x582a,0x65a4,0x5c48,0x604b,0x912d,0x5acc,
0x59ff,0x7159,0x502b,0x57cb,0x6416,0x5893,0x8d6b,0x901d,
0x5c82,0x75f4,0x699c,0x7e54,0x8058,0x676f,0x6e9c,0x6349,
0x4fa0,0x7ffc,0x8fdf,0x6f0f,0x6316,0x6676,0x60a3,0x7f29,
0x51f6,0x8f9e,0x9f84,0x8907,0x5f84,0x5de1,0x8d56,0x838a,
0x4e59,0x66f0,0x6124,0x5b99,0x60e8,0x63a9,0x82d7,0x5bf8,
0x9ea6,0x83e9,0x64fe,0x63da,0x5782,0x817e,0x9038,0x55b5,
0x54fc,0x7b51,0x7661,0x66fc,0x983b,0x67ef,0x5c97,0x7fc1,
0x94fa,0x91dd,0x71c3,0x6321,0x8de8,0x66ab,0x6d82,0x886b,
0x9670,0x5ef7,0x4ed7,0x6bb7,0x526a,0x5e10,0x8fa8,0x67f4,
0x7a00,0x6f20,0x52ff,0x732a,0x5915,0x7626,0x8179,0x8d74,
0x8fa3,0x529d,0x7b4b,0x87f2,0x71d2,0x6572,0x75c7,0x5112,
0x9801,0x8a93,0x79d2,0x52ab,0x6324,0x856d,0x543e,0x6590,
0x6d01,0x5be7,0x51fd,0x61b2,0x708e,0x5974,0x64a5,0x9e23,
0x5ac1,0x9676,0x5c38,0x626e,0x9aee,0x6fb3,0x5f6c,0x6cf3,
0x9897,0x9f9c,0x7fbd,0x5f6a,0x8389,0x984f,0x64a4,0x9592,
0x4e27,0x6b98,0x7267,0x5582,0x76d2,0x8205,0x61be,0x8017,
0x57c3,0x540a,0x6247,0x6296,0x70c2,0x9d3b,0x871c,0x9875,
0x96d5,0x8faf,0x796d,0x64ec,0x9055,0x5c18,0x6bbf,0x6182,
0x68af,0x996e,0x6d3d,0x5c4f,0x4f8d,0x52de,0x5be2,0x7fe0,
0x65e8,0x7eea,0x6daf,0x52c9,0x6d8c,0x6236,0x7fd4,0x7433,
0x984d,0x8d3c,0x64fa,0x9006,0x6a6b,0x53db,0x6127,0x5e9f,
0x556a,0x8be6,0x6c64,0x5f7b,0x758f,0x8f70,0x6328,0x9f4a,
0x601c,0x5f26,0x68da,0x8cfa,0x6efe,0x62ab,0x9e1f,0x85aa,
0x806a,0x8fa9,0x6bc1,0x8b5c,0x866b,0x997f,0x6109,0x70cf,
0x77ad,0x5339,0x67ab,0x9b06,0x6c1b,0x97ad,0x9640,0x6323,
0x75b2,0x5783,0x5a1c,0x5f2f,0x80ce,0x5687,0x6db2,0x87f9,
0x9b27,0x573e,0x52fe,0x848b,0x5203,0x75d5,0x5b6b,0x7199,
0x8fdd,0x4e8f,0x6b20,0x7260,0x641c,0x59a5,0x820c,0x4e22,

0x9396,0x51f1,0x640d,0x67c4,0x5951,0x7ed5,0x4e10,0x679a,
0x6dfb,0x4e11,0x6682,0x8070,0x73ab,0x614c,0x9012,0x6dd1,
0x8ff4,0x7af6,0x8702,0x60f9,0x5448,0x53b2,0x9e3f,0x715e,
0x6de8,0x901b,0x727d,0x621a,0x888d,0x95f7,0x5496,0x611a,
0x6e34,0x52b1,0x8230,0x5f70,0x5085,0x7275,0x7483,0x98c4,
0x4e18,0x7235,0x6367,0x6021,0x6dfa,0x59fb,0x7802,0x5851,
0x65a5,0x737b,0x75af,0x9eb5,0x541e,0x8266,0x5821,0x607c,
0x7e31,0x6016,0x9583,0x98a4,0x84ee,0x73bb,0x9ea5,0x6bc5,
0x95b1,0x8ad2,0x7cd6,0x5351,0x52a3,0x5be9,0x6bc0,0x6674,
0x53ed,0x6291,0x8270,0x7470,0x95c6,0x73ca,0x6191,0x94bb,
0x5561,0x93ae,0x8870,0x5ed6,0x90c1,0x8877,0x6168,0x50ac,
0x732b,0x7a79,0x6d9b,0x5146,0x92d2,0x7cd5,0x5bec,0x64ce,
0x6602,0x9505,0x62f3,0x7891,0x614e,0x9a45,0x5353,0x7f5a,
0x76c6,0x6d53,0x952e,0x8109,0x90bb,0x9501,0x9817,0x9063,
0x59ae,0x81e5,0x6b6a,0x5507,0x524a,0x9a7e,0x7978,0x9059,
0x880d,0x5967,0x9f4b,0x51a4,0x69cd,0x8096,0x89c8,0x9589,
0x62d3,0x5751,0x9813,0x810f,0x77ee,0x8180,0x6863,0x52f5,
0x629b,0x8679,0x9a71,0x7a9d,0x88e4,0x543b,0x9614,0x6dcb,
0x8a2a,0x655d,0x739b,0x9891,0x7985,0x7f69,0x85a9,0x98f2,
0x95a3,0x7838,0x5c1d,0x4ea8,0x7c92,0x576a,0x68cd,0x76d7,
0x76f2,0x5deb,0x7b79,0x964c,0x6436,0x5be1,0x77ac,0x6d45,
0x5154,0x53c9,0x778e,0x7dd2,0x77e9,0x4e0c,0x5490,0x5118,
0x6383,0x62bc,0x6b04,0x5617,0x6b96,0x538c,0x52f8,0x72c4,
0x7ff0,0x8d81,0x800d,0x8dea,0x9bae,0x7092,0x596e,0x8cdc,
0x5764,0x95e1,0x9274,0x8d8b,0x64f4,0x6ce3,0x742a,0x54aa,
0x8d2f,0x5d14,0x62e8,0x900a,0x8154,0x76dc,0x8482,0x8d54,
0x7f70,0x8c6b,0x67af,0x6495,0x7cb9,0x846c,0x68c9,0x88ad,
0x54ce,0x9a76,0x60e7,0x7ebd,0x54c9,0x76e3,0x8e2a,0x7c4c,
0x4e1b,0x9072,0x9510,0x8a87,0x8776,0x7a05,0x6e7f,0x7741,
0x77e3,0x7f50,0x7d1b,0x6746,0x6d51,0x8d62,0x5a36,0x9a70,
0x6052,0x70e4,0x8a95,0x9b31,0x7832,0x5996,0x7246,0x9970,
0x7f38,0x7aa9,0x507f,0x50be,0x7f20,0x8fad,0x6756,0x6d74,
0x62d8,0x6254,0x6444,0x6876,0x62df,0x6208,0x8f1b,0x6d12,
0x9f61,0x95ef,0x7b52,0x5026,0x8fb0,0x745c,0x716e,0x813e,
0x9971,0x7f1d,0x908f,0x8c48,0x6dbc,0x6b47,0x7378,0x79e9,
0x5f17,0x54a6,0x84c4,0x5f91,0x70bc,0x6f5b,0x5baa,0x5e99,
0x8292,0x8155,0x50a8,0x535c,0x51af,0x5524,0x7336,0x8d2a,
0x906e,0x6270,0x5367,0x7bc9,0x53ee,0x8f9c,0x5442,0x80c1,
0x5bb0,0x5a49,0x7fc5,0x5674,0x6591,0x68f5,0x5a66,0x8c31,
0x5e63,0x638f,0x5984,0x58ef,0x8e29,0x99a8,0x6deb,0x9601,
0x532a,0x6614,0x7164,0x9e4f,0x9b6f,0x5104,0x59e5,0x5362,
0x82af,0x54bd,0x6065,0x7efc,0x7b1b,0x5352,0x6368,0x9b45,
0x7779,0x6655,0x633d,0x8247,0x62e6,0x6e6f,0x6014,0x7aae,
0x52c1,0x745f,0x6b67,0x67dc,0x522e,0x77aa,0x6f06,0x81bd,
0x96fe,0x919c,0x5c3f,0x8e8d,0x7e73,0x7f55,0x5319,0x5bb4,
0x803b,0x8086,0x644a,0x5835,0x97d3,0x5f65,0x99d5,0x8822,
0x54b3,0x7de9,0x6012,0x8bde,0x6846,0x60ac,0x634f,0x7334,
0x537f,0x71ac,0x6046,0x4f51,0x6789,0x552c,0x51d1,0x80c3,
0x5e15,0x964b,0x55bb,0x8ed2,0x5492,0x5589,0x60f6,0x5a9a,
0x8299,0x541d,0x7b3c,0x98a0,0x5f4e,0x5288,0x643a,0x5537,
0x8ce6,0x6cc4,0x809d,0x754f,0x63b7,0x5429,0x522a,0x7ea0,

0x66ae,0x7919,0x7a23,0x76c3,0x82b7,0x8d9f,0x96c0,0x9739,
0x55e8,0x5428,0x62c2,0x6fc3,0x64cb,0x53a8,0x7ef3,0x88f9,
0x91e3,0x56b7,0x905c,0x6da8,0x76b1,0x8d4c,0x5993,0x7aed,
0x8116,0x77ff,0x5c39,0x4f10,0x90ca,0x7545,0x819d,0x54c4,
0x5938,0x5b55,0x55b7,0x5606,0x9556,0x8e5f,0x4ec6,0x5f0a,
0x6491,0x60f1,0x76ef,0x63a0,0x7089,0x88d9,0x59e8,0x60df,
0x6ec4,0x80a2,0x962e,0x8523,0x4f2a,0x6f54,0x4ffa,0x8c05,
0x596a,0x80a0,0x9493,0x840a,0x8caa,0x5265,0x6284,0x8de1,
0x8d5a,0x937e,0x7a4c,0x5320,0x96c1,0x62da,0x6fa1,0x5e16,
0x56ca,0x70db,0x7642,0x790e,0x50d1,0x6db5,0x9727,0x8fc8,
0x5fb9,0x9f7f,0x8b00,0x5d16,0x8c28,0x8258,0x4e19,0x72a7,
0x7e5e,0x7529,0x5f25,0x58fd,0x723a,0x9a37,0x5378,0x64d2,
0x502a,0x5e06,0x808c,0x7e6a,0x98fd,0x9cf4,0x503a,0x6627,
0x86d9,0x8f9f,0x5239,0x8ca2,0x62d0,0x80a4,0x96c7,0x5495,
0x58e2,0x8e72,0x4fef,0x543c,0x8e48,0x8bf1,0x64bf,0x9b44,
0x8015,0x9716,0x798d,0x7554,0x5925,0x60a6,0x8273,0x6490,
0x6372,0x5d50,0x5632,0x8d37,0x5401,0x752b,0x9742,0x64e0,
0x89bd,0x5bd3,0x8b6c,0x746a,0x9888,0x9b41,0x8df5,0x55a7,
0x658c,0x8d3e,0x632a,0x8f7f,0x6df9,0x51a5,0x5a07,0x5c65,
0x971c,0x6d78,0x6f47,0x6dd8,0x9326,0x8d50,0x6953,0x8cd3,
0x575f,0x515c,0x9882,0x5021,0x7344,0x5074,0x5937,0x7816,
0x6e14,0x5b9b,0x5c51,0x60b6,0x5bb5,0x5dfe,0x6b79,0x900d,
0x5ac2,0x6380,0x9709,0x54d1,0x55ac,0x9f52,0x997c,0x632b,
0x4fae,0x82ad,0x5dba,0x97fb,0x6ee9,0x727a,0x9489,0x88f8,
0x8e64,0x8d60,0x94c3,0x9081,0x7a1a,0x50a2,0x987d,0x6795,
0x8b7d,0x5a1f,0x8932,0x5d29,0x902e,0x50f5,0x916c,0x79e4,
0x8f68,0x54df,0x99db,0x7855,0x6734,0x78d5,0x5e05,0x61d2,
0x819c,0x517d,0x51c4,0x8a79,0x5583,0x730e,0x6500,0x574e,
0x9965,0x6bbc,0x5ab3,0x55d3,0x5eff,0x5147,0x934b,0x903b,
0x61fc,0x92b3,0x508d,0x8d29,0x9a84,0x84b8,0x7ed8,0x96ef,
0x7109,0x7948,0x64b0,0x4e2b,0x8667,0x604d,0x6670,0x95ca,
0x5857,0x9119,0x593e,0x9130,0x6085,0x4fde,0x8f5f,0x86ee,
0x640f,0x99d0,0x6398,0x8ced,0x4fd8,0x8350,0x62f7,0x8eac,
0x6b3d,0x6d29,0x51f3,0x9905,0x5f4c,0x8cab,0x9017,0x72ac,
0x7db1,0x7ff9,0x65a9,0x50da,0x58ae,0x9811,0x5415,0x8c79,
0x6ea2,0x8d08,0x8461,0x5f77,0x722a,0x6055,0x5c2c,0x6f8e,
0x62cb,0x5c4e,0x68ad,0x8c2d,0x683d,0x6467,0x584c,0x788c,
0x68fa,0x57ae,0x5824,0x51bb,0x79aa,0x9e9f,0x707d,0x9a86,
0x5e18,0x5075,0x6cfc,0x62f1,0x88d4,0x8b9a,0x6726,0x6292,
0x8404,0x8c26,0x5c09,0x5395,0x8da8,0x7737,0x4e5e,0x8e81,
0x9077,0x814a,0x9edb,0x7210,0x617e,0x9c8d,0x8650,0x4ed3,
0x79c3,0x5a77,0x7ed1,0x672d,0x764c,0x8235,0x803f,0x755c,
0x60bc,0x9e2d,0x7184,0x6feb,0x6f32,0x8bca,0x8ce4,0x5466,
0x54e8,0x7eb9,0x9d5d,0x5e9e,0x8ecc,0x9a9a,0x5858,0x55e4,
0x8c9e,0x895f,0x4f84,0x7955,0x8766,0x57d4,0x8b20,0x81a0,
0x905e,0x4f6c,0x54d7,0x69fd,0x4ea9,0x9ad2,0x6e20,0x561f,
0x8c0e,0x5006,0x7bee,0x88d8,0x6401,0x5cfb,0x53a2,0x868a,
0x5ae3,0x5faa,0x7792,0x6c90,0x5c4d,0x947d,0x56da,0x80bf,
0x810a,0x6c13,0x7830,0x5564,0x8fe6,0x5ec1,0x9db4,0x55aa,
0x4fa8,0x53e0,0x7051,0x75ab,0x5578,0x64c5,0x80c0,0x8d4b,
0x52df,0x8108,0x6073,0x7e8f,0x5a74,0x8e44,0x8165,0x714e,

0x664c,0x6afb,0x6e3e,0x6ed4,0x6bd9,0x7329,0x6963,0x5641,
0x8102,0x8c1c,0x6c27,0x8774,0x857e,0x545c,0x5bc7,0x6233,
0x9881,0x7a9c,0x884d,0x5132,0x687f,0x7942,0x7e6b,0x988a,
0x5140,0x8332,0x5631,0x9127,0x68df,0x9a73,0x9a30,0x59ec,
0x7cbd,0x53e1,0x5662,0x4f75,0x5243,0x80ba,0x9eef,0x566a,
0x6649,0x6487,0x9d28,0x4f83,0x6e3a,0x6caa,0x66a2,0x8d31,
0x90dd,0x7130,0x5291,0x56bc,0x602f,0x98c6,0x651c,0x7b77,
0x5992,0x87ba,0x83cc,0x58c7,0x559a,0x94a9,0x5102,0x7ad6,
0x60e9,0x803d,0x6eb6,0x7ee3,0x90e1,0x8bb6,0x7eb2,0x6ac3,
0x87fb,0x8ca9,0x8231,0x62d9,0x5a31,0x9e26,0x72ee,0x560e,
0x9699,0x7a9f,0x74f7,0x51db,0x9f9f,0x7fa1,0x711a,0x903e,
0x7a91,0x8972,0x8587,0x5ba0,0x7ea4,0x94fe,0x7aff,0x8b39,
0x853d,0x655e,0x72ed,0x6558,0x4f3a,0x94ed,0x8cca,0x86db,
0x8fc4,0x68b3,0x6e0a,0x83bd,0x71e6,0x7faf,0x4fb6,0x6524,
0x886c,0x7a57,0x7fa8,0x8b0e,0x7dbf,0x7f05,0x7c9e,0x8ce0,
0x6402,0x918b,0x7c64,0x549a,0x533f,0x9a87,0x8105,0x50fb,
0x761f,0x6bcb,0x81a8,0x7f1a,0x547b,0x707f,0x7d10,0x8206,
0x55b2,0x8a60,0x818f,0x5375,0x83c1,0x65a7,0x58f9,0x6514,
0x97f5,0x7a83,0x819a,0x9a5f,0x55e1,0x66a8,0x80f3,0x748b,
0x6123,0x7693,0x9877,0x75de,0x673d,0x6cb8,0x5308,0x5edf,
0x63ea,0x6687,0x4fa6,0x6577,0x8178,0x75d2,0x7fe9,0x6346,
0x8038,0x9171,0x65f1,0x5009,0x58f6,0x9661,0x5bee,0x8b0a,
0x6652,0x8cbf,0x9a74,0x58f3,0x6ca7,0x79a6,0x8d1e,0x818a,
0x8e34,0x6bef,0x860b,0x71e5,0x5b7d,0x64bc,0x4fcf,0x8f85,
0x79be,0x7538,0x595a,0x8511,0x5d17,0x8d26,0x76ea,0x5be5,
0x66c6,0x8403,0x532f,0x8a98,0x5366,0x557c,0x6361,0x60ed,
0x599e,0x5636,0x553e,0x8b19,0x7caa,0x9470,0x9215,0x6ee5,
0x5315,0x582f,0x76e7,0x6e83,0x9a7c,0x96b6,0x61a4,0x879e,
0x4e52,0x8bc0,0x65f7,0x5962,0x7d0b,0x744b,0x56c2,0x5256,
0x5c34,0x6bd3,0x70ad,0x7f34,0x5f7f,0x5450,0x7375,0x88b1,
0x52f3,0x82f9,0x61c7,0x74ca,0x51cd,0x8782,0x78ca,0x7a4e,
0x8ef8,0x540b,0x5514,0x5986,0x6dc0,0x8721,0x58e4,0x851a,
0x6a11,0x5c61,0x6273,0x6f51,0x51f8,0x970e,0x9f90,0x63e3,
0x5242,0x8c41,0x7c98,0x608d,0x9285,0x9aa4,0x95a9,0x6177,
0x8be7,0x7cfe,0x53e2,0x819b,0x5e62,0x8f96,0x55fd,0x934a,
0x6e4a,0x960e,0x7ca5,0x85b0,0x87d1,0x63b0,0x62e2,0x7a3c,
0x8463,0x7784,0x7728,0x80e7,0x6d95,0x7977,0x8bbd,0x9ecf,
0x6583,0x94ee,0x9a55,0x5983,0x79c9,0x511f,0x60d5,0x62e3,
0x9187,0x78b3,0x84e6,0x6869,0x540f,0x8569,0x6f64,0x8c23,
0x695e,0x5cb1,0x9913,0x5760,0x6ede,0x7011,0x7095,0x4f47,
0x7504,0x8bf5,0x659f,0x85af,0x6fd5,0x4f36,0x852c,0x75a4,
0x507d,0x8e10,0x7eb1,0x8ca7,0x8b2c,0x7a3d,0x83b9,0x8854,
0x8be1,0x817b,0x8d2c,0x99c1,0x6df5,0x717d,0x99b3,0x635e,
0x6405,0x8098,0x4f1e,0x94f8,0x8eaf,0x7b19,0x73c2,0x6eaf,
0x70b3,0x65ac,0x63c9,0x6b7c,0x8a6d,0x82bd,0x6631,0x8042,
0x6dcc,0x5fff,0x9980,0x70eb,0x821c,0x4ed1,0x949e,0x77a5,
0x6d46,0x72f8,0x5c94,0x4ed5,0x625b,0x8543,0x6893,0x5a7f,
0x7c60,0x7bf7,0x5960,0x8a1d,0x6666,0x985b,0x6cae,0x745b,
0x8335,0x7bab,0x64b2,0x776c,0x9e45,0x917f,0x53e8,0x572d,
0x7662,0x8328,0x8a57,0x6e85,0x5179,0x6363,0x5029,0x5431,
0x680b,0x6da9,0x7980,0x7f06,0x82df,0x70c1,0x8bc8,0x61ff,

```
 0x6e1d,0x6026,0x8db4,0x8d66,0x53ae,0x631f,0x51b6,0x6115,
 0x8cc4,0x9ad3,0x7dfb,0x9068,0x72e1,0x542d,0x88f3,0x9952,
 0x7426,0x82db,0x6d85,0x6413,0x7422,0x95d6,0x8af7,0x9791,
 0x8a3a,0x8ae7,0x6399,0x921e,0x8993,0x701f,0x8654,0x7eba,
 0x68a2,0x92ea,0x61f8,0x70ab,0x9524,0x4e53,0x5a34,0x5151,
 0x8a6e,0x51f9,0x8fab,0x6c41,0x650f,0x58fa,0x9cc4,0x76cf,
 0x4e4d,0x7115,0x7076,0x5815,0x6dd2,0x7c3f,0x674f,0x89c5,
 0x8085,0x8c10,0x6f13,0x5955,0x70d8,0x6ffe,0x7e96,0x91ac,
 0x7d6e,0x618e,0x5885,0x5e9a,0x8f3b,0x79a7,0x94a5,0x634d,
 0x5dcd,0x7eee,0x9713,0x79bd,0x62ef,0x66dd,0x5bc5,0x5ac9,
 0x5bf5,0x60b8,0x6f01,0x5384,0x588a,0x7ef8,0x5201,0x7e2b,
 0x99ff,0x763e,0x7736,0x53e9,0x901e,0x8e66,0x6020,0x57ab,
 0x6d47,0x5f64,0x60d8,0x52d8,0x5f8a,0x8046,0x618b,0x95f5,
 0x99dd,0x8549,0x50d5,0x758a,0x62ed,0x55dc,0x7b8f,0x82b8,
 0x7194,0x524e,0x840e,0x589c,0x6912,0x8513,0x592d,0x766e,
 0x7efd,0x881f,0x5f98,0x725f,0x96a7,0x68b5,0x79b1,0x772f,
 0x5162,0x6a61,0x914c,0x749e,0x7c72,0x5f13,0x5a1b,0x98b1,
 0x9798,0x7409,0x773a,0x64ab,0x9e20,0x5098,0x5b0c,0x932b,
 0x77bb,0x6c85,0x6ca6,0x7f15,0x889c,0x6c8c,0x797a,0x79bf,
 0x6dea,0x7682,0x525d,0x759a,0x841d,0x9ae6,0x795f,0x6f9c,
 0x8700,0x53a5,0x6e67,0x6e17,0x500f,0x7a98,0x61c8,0x6043,
 0x63fd,0x82b9,0x8f69,0x6cd3,0x836b,0x7d43,0x78da,0x6c83
};

#define UNIHAN_REORDER_BASE 0X5000

u_code_point reorder_unihan(u_code_point s) {
        u_code_point i=UNIHAN_REORDER_BASE;
        int k=0;
        for(k=0; k<UH; k++,i++) {
           if(s == unihan_freq[k]) { return i; };
        };
        k=(s - UNIHAN_REORDER_BASE);
        if( k>=0 && k<UH) {
           return unihan_freq[k];
        };
        return s;
}

u_code_point restore_order_unihan(u_code_point z) {
        u_code_point i=UNIHAN_REORDER_BASE;
        int k;
        k=(z - UNIHAN_REORDER_BASE);
        if( k>=0 && k<UH) {
           return unihan_freq[k];
        };
        for(k=0; k<UH; k++,i++) {
           if(z == unihan_freq[k]) { return i; };
        };
        return z;
}
```

```
#define KATAKANA_REORDER_BASE 0X30A0
//KATAKANA reorder by frequency in Japanese Business
#define KK 96
u_code_point katakana_freq[KK] = {
0x30f3,0x30eb,0x30b9,0x30c8,0x30a2,0x30a4,0x30e9,0x30ea,
0x30af,0x30c3,0x30fc,0x30b7,0x30b8,0x30e7,0x30ec,0x30b0,
0x30d5,0x30d7,0x30df,0x30c4,0x30ef,0x30a8,0x30cb,0x30e1,
0x30ab,0x30c6,0x30b3,0x30dd,0x30d9,0x30cf,0x30c9,0x30a6,
0x30bb,0x30ce,0x30ca,0x30e0,0x30ed,0x30bf,0x30c1,0x30d0,
0x30b4,0x30dc,0x30bd,0x30cd,0x30e2,0x30d3,0x30b5,0x30ad,
0x30b1,0x30b2,0x30bc,0x30e8,0x30e5,0x30aa,0x30cc,0x30a3,
0x30d6,0x30de,0x30a1,0x30a5,0x30a7,0x30a9,0x30ac,0x30ae,
0x30b6,0x30ba,0x30be,0x30c0,0x30c2,0x30c5,0x30c7,0x30d1,
0x30d2,0x30d4,0x30d8,0x30da,0x30db,0x30e3,0x30e4,0x30e6,
0x30ee,0x30f0,0x30f1,0x30f2,0x30f4,0x30f5,0x30f6,0x30f7,
0x30f8,0x30f9,0x30fa,0x30fb,0x30fd,0x30fe,0x30ff,0x30a0,
};

u_code_point reorder_katakana(u_code_point s) {
        u_code_point i=KATAKANA_REORDER_BASE;
        int k=0;
        for(k=0; k<KK; k++,i++) {
           if(s == katakana_freq[k]) { return i; };
        };
        return s; // not reached here
}

u_code_point restore_order_katakana(u_code_point z) {
        return katakana_freq[z - KATAKANA_REORDER_BASE];
}


#define HINDI_REORDER_BASE 0X0900
//HINDI reorder by frequency
#define HD 113
u_code_point hindi_freq[HD] = {
0x0902,0x093f,0x0940,0x0947,0x0948,0x094b,0x094d,0x0915,
0x0917,0x0932,0x0938,0x0939,0x0924,0x0926,0x0928,0x092c,
0x092f,0x0900,0x0901,0x0903,0x0904,0x0916,0x0918,0x0919,
0x091a,0x091b,0x091c,0x091d,0x091e,0x091f,0x0920,0x0921,
0x0922,0x0923,0x0925,0x0927,0x0929,0x092a,0x092b,0x092d,
0x092e,0x0930,0x0931,0x0933,0x0934,0x0935,0x0936,0x0937,
0x093a,0x093b,0x093c,0x093d,0x093e,0x0941,0x0942,0x0943,
0x0944,0x0945,0x0946,0x0949,0x094a,0x094c,0x094e,0x094f,
0x0950,0x0951,0x0952,0x0953,0x0954,0x0955,0x0956,0x0957,
0x0958,0x0959,0x095a,0x095b,0x095c,0x095d,0x095e,0x095f,
0x0960,0x0961,0x0962,0x0963,0x0964,0x0965,0x0966,0x0967,
0x0968,0x0969,0x096a,0x096b,0x096c,0x096d,0x096e,0x096f,
0x0905,0x0906,0x0907,0x0908,0x0909,0x090a,0x090b,0x090c,
```

```
0x090d,0x090e,0x090f,0x0910,0x0911,0x0912,0x0913,0x0914,
0x0970
};

u_code_point reorder_hindi(u_code_point s) {
        u_code_point i=HINDI_REORDER_BASE;
        int k=0;
        for(k=0; k<HD; k++,i++) {
           if(s == hindi_freq[k]) { return i; };
        };
        return s; // not reached here
}

u_code_point restore_order_hindi(u_code_point z) {
        return hindi_freq[z - HINDI_REORDER_BASE];
}


#define MAPCHAR(x,A,B,bytes) if(A<=x && x< (A+bytes)) \
        return(x+(B-A)); if(B<=x && x< (B+bytes)) return(x+(A-B))
#define MAP16BL(x,A,B,block) if(A<=x && x< (A+(block<<4))) \
        return(x+(B-A)); if(B<=x && x< (B+(block<<4))) \
        return(x+(A-B))

u_code_point reorder_latins(u_code_point s) {
        MAP16BL(s,0x0100,0x0000,3); // Latin Extension A
        MAP16BL(s,0x0130,0x0080,2);
        MAP16BL(s,0x0150,0x00A0,1);
        MAP16BL(s,0x0300,0x00B0,3); // Combining Diacritical Marks
        MAPCHAR(s,0x0070,0x0060,1); // p,`
        MAPCHAR(s,0x0072,0x006A,1); // r,j
        MAPCHAR(s,0x0073,0x006B,1); // s,k
        MAPCHAR(s,0x0074,0x0066,1); // t,f
        MAPCHAR(s,0x0075,0x0067,1); // u,g
        MAPCHAR(s,0x0050,0x0040,1); // P,@  UPPER
        MAPCHAR(s,0x0052,0x004A,1); // R,J  UPPER
        MAPCHAR(s,0x0053,0x004B,1); // S,K  UPPER
        MAPCHAR(s,0x0054,0x0046,1); // T,F  UPPER
        MAPCHAR(s,0x0055,0x0047,1); // U,G  UPPER
        MAPCHAR(s,0x0160,0x003A,6); // Latin Extension A
        MAPCHAR(s,0x0166,0x005B,5);
        MAPCHAR(s,0x016B,0x007B,5);
        return s;
}

u_code_point restore_order_latins(u_code_point z) {
        return reorder_latins(z);
}

u_code_point reorder(u_code_point s) {
        if(isHANGUL(s)) return reorder_hangul(s);
```

```
        if(isUNIHAN(s)) return reorder_unihan(s);
        if(isKATAKANA(s)) return reorder_katakana(s);
        if(isHINDI(s)) return reorder_hindi(s);
        if(isLatins(s)) return reorder_latins(s);
        return s;
}
u_code_point restore_order(u_code_point s) {
        if(isHANGUL(s)) return restore_order_hangul(s);
        if(isUNIHAN(s)) return restore_order_unihan(s);
        if(isKATAKANA(s)) return restore_order_katakana(s);
        if(isHINDI(s)) return restore_order_hindi(s);
        if(isLatins(s)) return restore_order_latins(s);
        return s;
}


/* Encoder: */

enum dude_status dude_encode(
  unsigned int input_length,
  const u_code_point input[],
  const unsigned char uppercase_flags[],
  unsigned int *output_size,
  char output[] )
{
  unsigned int max_out, in, out, k, j;
  u_code_point prev, codept, diff, tmp;
  char shift;

  prev = 0x60;
  //prev = 0x5000;
  max_out = *output_size;

  for (in = out = 0;  in < input_length;  ++in) {

    /* At the start of each iteration, in and out are the number of */
    /* items already input/output, or equivalently, the indices of  */
    /* the next items to be input/output.                           */

    codept = input[in];

    if (codept == 0x2D) {
      /* Hyphen-minus stands for itself. */
      if (max_out - out < 1) return dude_big_output;
      output[out++] = 0x2D;
      continue;
    }

    codept = reorder(codept); // by LSB

    diff = prev^codept;
```

```
    /* Compute the number of base-32 characters (k): */
    for (tmp = diff >> 4, k = 1;  tmp != 0;  ++k, tmp >>= 4);
    fprintf(stderr,"diff %x,%x = prev %x ^ codept %x \n",
        k,diff,prev,codept);

    if (max_out - out < k) return dude_big_output;
    shift = uppercase_flags && uppercase_flags[in] ? 32 : 0;
    /* shift controls the case of the last base-32 digit. */

    /* Each quintet has the form 1xxxx except the last is 0xxxx. */
    /* Computing the base-32 digits in reverse order is easiest. */

    out += k;
    output[out - 1] = base32[diff & 0xF] - shift;

    for (j = 2;  j <= k;  ++j) {
      diff >>= 4;
      output[out - j] = base32[0x10 | (diff & 0xF)];
    }

    prev = codept;
  }

  /* Append the null terminator: */
  if (max_out - out < 1) return dude_big_output;
  output[out++] = 0;

  *output_size = out;
  return dude_success;
}


/* Decoder: */

enum dude_status dude_decode(
  enum case_sensitivity case_sensitivity,
  char scratch_space[],
  const char input[],
  unsigned int *output_length,
  u_code_point output[],
  unsigned char uppercase_flags[] )
{
  u_code_point prev, q, diff;
  char c;
  unsigned int max_out, in, out, scratch_size;
  enum dude_status status;

  prev = 0x60;
  max_out = *output_length;

  for (c = input[in = 0], out = 0;  c != 0;  c = input[++in], ++out) {
```

```
    /* At the start of each iteration, in and out are the number of */
    /* items already input/output, or equivalently, the indices of  */
    /* the next items to be input/output.                           */

    if (max_out - out < 1) return dude_big_output;

    if (c == 0x2D) output[out] = c;  /* hyphen-minus is literal */
    else {
      /* Base-32 sequence.  Decode quintets until 0xxxx is found: */

      for (diff = 0;  ;  c = input[++in]) {
        q = base32_decode(c);
        if (q == base32_invalid){ return dude_bad_input; };
        diff = (diff << 4) | (q & 0xF);
        if (q >> 4 == 0) break;
      }

      // prev = output[out] = prev ^ diff;
      prev = prev ^ diff;
      output[out] = restore_order(prev); // LSB

    }

    /* Case of last character determines uppercase flag: */
    if (uppercase_flags) uppercase_flags[out] = c >= 65 && c <= 90;
  }

  /* Enforce the uniqueness of the encoding by re-encoding */
  /* the output and comparing the result to the input:     */

  scratch_size = ++in;
  status = dude_encode(out, output, uppercase_flags,
                       &scratch_size, scratch_space);
  if (status != dude_success || scratch_size != in ||
      unequal(case_sensitivity, scratch_space, input)
     ) return dude_bad_input;
  *output_length = out;
  return dude_success;
}


/*****************************************************************/
/* Wrapper for testing (would normally go in a separate .c file): */

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* For testing, we'll just set some compile-time limits rather than */
/* use malloc(), and set a compile-time option rather than using a  */
```

```
/* command-line option.                                              */

enum {
  unicode_max_length = 256,
  ace_max_size = 256,
  test_case_sensitivity = case_insensitive
                          /* suitable for host names */
};


static void usage(char **argv)
{
  fprintf(stderr,
    "%s -e reads code points and writes a DUDE string.\n"
    "%s -d reads a DUDE string and writes code points.\n"
    "Input and output are plain text in the native character set.\n"
    "Code points are in the form u+hex separated by whitespace.\n"
    "A DUDE string is a newline-terminated sequence of LDH characters\n"
    "(without any signature).\n"
    "The case of the u in u+hex is the force-to-uppercase flag.\n"
    , argv[0], argv[0]);
  exit(EXIT_FAILURE);
}


static void fail(const char *msg)
{
  fputs(msg,stderr);
  exit(EXIT_FAILURE);
}

static const char too_big[] =
  "input or output is too large, recompile with larger limits\n";
static const char invalid_input[] = "invalid input\n";
static const char io_error[] = "I/O error\n";


/* The following string is used to convert LDH      */
/* characters between ASCII and the native charset: */

static const char ldh_ascii[] =
  "................"
  "................"
  "............-.."
  "0123456789......"
  ".ABCDEFGHIJKLMNO"
  "PQRSTUVWXYZ....."
  ".abcdefghijklmno"
  "pqrstuvwxyz";


int main(int argc, char **argv)
```

```
{
  enum dude_status status;
  int r;
  char *p;

  if (argc != 2) usage(argv);
  if (argv[1][0] != '-') usage(argv);
  if (argv[1][2] != 0) usage(argv);

  if (argv[1][1] == 'e') {
    u_code_point input[unicode_max_length];
    unsigned long codept;
    unsigned char uppercase_flags[unicode_max_length];
    char output[ace_max_size], uplus[3];
    unsigned int input_length, output_size, i;

    /* Read the input code points: */

    input_length = 0;

    for (;;) {
      r = scanf("%2s%lx", uplus, &codept);
      if (ferror(stdin)) fail(io_error);
      if (r == EOF || r == 0) break;

      if (r != 2 || uplus[1] != '+' || codept > (u_code_point)-1) {
        fail(invalid_input);
      }

      if (input_length == unicode_max_length) fail(too_big);

      if (uplus[0] == 'u') uppercase_flags[input_length] = 0;
      else if (uplus[0] == 'U') uppercase_flags[input_length] = 1;
      else fail(invalid_input);

      input[input_length++] = codept;
    }

    /* Encode: */

    output_size = ace_max_size;
    status = dude_encode(input_length, input, uppercase_flags,
                         &output_size, output);
    if (status == dude_bad_input) fail(invalid_input);
    if (status == dude_big_output) fail(too_big);
    assert(status == dude_success);

    /* Convert to native charset and output: */

    for (p = output;  *p != 0;  ++p) {
      i = *p;
      assert(i <= 122 && ldh_ascii[i] != '.');
```

```
    *p = ldh_ascii[i];
  }

  r = puts(output);
  fprintf(stderr,"length: %d\n", strlen(output));
  if (r == EOF) fail(io_error);
  return EXIT_SUCCESS;
}

if (argv[1][1] == 'd') {
  char input[ace_max_size], scratch[ace_max_size], *pp;
  u_code_point output[unicode_max_length];
  unsigned char uppercase_flags[unicode_max_length];
  unsigned int input_length, output_length, i;

  /* Read the DUDE input string and convert to ASCII: */

  fgets(input, ace_max_size, stdin);
  if (ferror(stdin)) fail(io_error);
  if (feof(stdin)) fail(invalid_input);
  input_length = strlen(input);
  if (input[input_length - 1] != '\n') fail(too_big);
  input[--input_length] = 0;

  for (p = input;  *p != 0;  ++p) {
    pp = strchr(ldh_ascii, *p);
    if (pp == 0) fail(invalid_input);
    *p = pp - ldh_ascii;
  }

  /* Decode: */

  output_length = unicode_max_length;
  status = dude_decode(test_case_sensitivity, scratch, input,
                       &output_length, output, uppercase_flags);
  if (status == dude_bad_input) fail(invalid_input);
  if (status == dude_big_output) fail(too_big);
  assert(status == dude_success);

  /* Output the result: */

  for (i = 0;  i < output_length;  ++i) {
    r = printf("%s+%04lX\n",
               uppercase_flags[i] ? "U" : "u",
               (unsigned long) output[i] );
    if (r < 0) fail(io_error);
  }

  return EXIT_SUCCESS;
}

usage(argv);
```

```
  return EXIT_SUCCESS;  /* not reached, but quiets compiler warning */
}

/* end of ldude.c */



LAMCW: Example implementation into AMC-ACE-W

    This idea is applicable to any ACEs.
    LAMCW is a name for AMC-ACE-W implementation of this idea.

    Embedded hangul,han and Latin frequency tables are subject
    to change with further studies in the next revision of this draft.

    In Unix, save this example source code into ldude.c

    % cc -o lamcw lamcw.c
    % ./lamcw -e < input_file > output_file
    % ./lamcw -d < output_file

    An input file should contains u+????-form code points
    delimited with spaces or newlines.

/* begin of lamcw.c */

/*****************************************************/
/* lamcw.c 1.0  (2001-Jul-3)                         */
/* Soobok Lee <lsb@postel.co.kr>                     */
/* amcw.c from Adam M. Costello <amc@cs.berkeley.edu> */
/*****************************************************/


/* This is ANSI C code (C89) implementing AMC-ACE-W version 0.1.*. */


/***********************************************************/
/* Public interface (would normally go in its own .h file): */

#include <limits.h>

enum amc_ace_status {
  amc_ace_success,
  amc_ace_bad_input,
  amc_ace_big_output  /* Output would exceed the space provided. */
};

enum case_sensitivity { case_sensitive, case_insensitive };

#if UINT_MAX >= 0x1FFFFF
typedef unsigned int u_code_point;
#else
```

```
typedef unsigned long u_code_point;
#endif

enum amc_ace_status amc_ace_w_encode(
  unsigned int input_length,
  const u_code_point input[],
  const unsigned char uppercase_flags[],
  unsigned int *output_size,
  char output[] );

    /* amc_ace_w_encode() converts Unicode to AMC-ACE-W (without      */
    /* any signature).  The input must be represented as an array     */
    /* of Unicode code points (not code units; surrogate pairs        */
    /* are not allowed), and the output will be represented as        */
    /* null-terminated ASCII.  The input_length is the number of      */
    /* code points in the input.  The output_size is an in/out        */
    /* argument: the caller must pass in the maximum number of        */
    /* characters that may be output (including the terminating       */
    /* null), and on successful return it will contain the number of  */
    /* characters actually output (including the terminating null,    */
    /* so it will be one more than strlen() would return, which is    */
    /* why it is called output_size rather than output_length).  The  */
    /* uppercase_flags array must hold input_length boolean values,   */
    /* where nonzero means the corresponding Unicode character should */
    /* be forced to uppercase after being decoded, and zero means it  */
    /* is caseless or should be forced to lowercase.  Alternatively,  */
    /* uppercase_flags may be a null pointer, which is equivalent     */
    /* to all zeros.  The letters a-z and A-Z are always encoded      */
    /* literally, regardless of the corresponding flags.  The encoder */
    /* always outputs lowercase base-32 characters except when        */
    /* nonzero values of uppercase_flags require otherwise.  The      */
    /* return value may be any of the amc_ace_status values defined   */
    /* above; if not amc_ace_success, then output_size and output may */
    /* contain garbage.  On success, the encoder will never need to   */
    /* write an output_size greater than input_length*5+1, because of */
    /* how the encoding is defined.                                   */

enum amc_ace_status amc_ace_w_decode(
  enum case_sensitivity case_sensitivity,
  char scratch_space[],
  const char input[],
  unsigned int *output_length,
  u_code_point output[],
  unsigned char uppercase_flags[] );

    /* amc_ace_w_decode() converts AMC-ACE-W (without any signature)  */
    /* to Unicode.  The input must be represented as null-terminated  */
    /* ASCII, and the output will be represented as an array of       */
    /* Unicode code points.  The case_sensitivity argument influences */
    /* the check on the well-formedness of the input string; it       */
    /* must be case_sensitive if case-sensitive comparisons are       */
```

```
    /* allowed on encoded strings, case_insensitive otherwise.        */
    /* The scratch_space must point to space at least as large        */
    /* as the input, which will get overwritten (this allows the      */
    /* decoder to avoid calling malloc()).  The output_length is      */
    /* an in/out argument: the caller must pass in the maximum        */
    /* number of code points that may be output, and on successful    */
    /* return it will contain the actual number of code points        */
    /* output.  The uppercase_flags array must have room for at       */
    /* least output_length values, or it may be a null pointer        */
    /* if the case information is not needed.  A nonzero flag         */
    /* indicates that the corresponding Unicode character should      */
    /* be forced to uppercase by the caller, while zero means it      */
    /* is caseless or should be forced to lowercase.  The letters     */
    /* a-z and A-Z are output already in the proper case, but their   */
    /* flags will be set appropriately so that applying the flags     */
    /* would be harmless.  The return value may be any of the         */
    /* amc_ace_status values defined above; if not amc_ace_success,   */
    /* then output_length, output, and uppercase_flags may contain    */
    /* garbage.  On success, the decoder will never need to write     */
    /* an output_length greater than the length of the input (not     */
    /* counting the null terminator), because of how the encoding is  */
    /* defined.                                                       */


/**********************************************************/
/* Implementation (would normally go in its own .c file): */

#include <string.h>

/* base32[q] is the lowercase base-32 character representing  */
/* the number q from the range 0 to 31.  Note that we cannot  */
/* use string literals for ASCII characters because an ANSI C */
/* compiler does not necessarily use ASCII.                   */

static const char base32[] = {
  97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,     /* a-k */
  109, 110,                                               /* m-n */
  112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,  /* p-z */
  50, 51, 52, 53, 54, 55, 56, 57                          /* 2-9 */
};

/* base32_decode(c) returns the value of a base-32 character, in the */
/* range 0 to 31, or the constant base32_invalid if c is not a valid */
/* base-32 character.                                                */

enum { base32_invalid = 32 };

static unsigned int base32_decode(char c)
{
  if (c < 50) return base32_invalid;
  if (c <= 57) return c - 26;
```

```
  if (c < 97) c += 32;
  if (c < 97 || c == 108 || c == 111 || c > 122) return base32_invalid;
  return c - 97 - (c > 108) - (c > 111);
}

/* unequal(case_sensitivity,s1,s2) returns 0 if the strings s1 and s2 */
/* are equal, 1 otherwise.  If case_sensitivity is case_insensitive,  */
/* then ASCII A-Z are considered equal to a-z respectively.           */

static int unequal( enum case_sensitivity case_sensitivity,
                    const char s1[], const char s2[]        )
{
  char c1, c2;

  if (case_sensitivity != case_insensitive) return strcmp(s1,s2) != 0;

  for (;;) {
    c1 = *s1;
    c2 = *s2;
    if (c1 >= 65 && c1 <= 90) c1 += 32;
    if (c2 >= 65 && c2 <= 90) c2 += 32;
    if (c1 != c2) return 1;
    if (c1 == 0) return 0;
    ++s1, ++s2;
  }
}



/* LANGUAGE-SPECIFIC IMPROVEMENTS TO DUDE  BASED ON CODE REORDERING */

int isHANGUL(u_code_point s) {
        int SIndex = s - 0xAC00;
        if (SIndex < 0 || SIndex >= 11172) {
            return 0;
        }
        return 1;
};
int isUNIHAN(u_code_point s) {
        if (s >= 0x4E00 && s <= 0x9FAF) {
            return 1;
        }
        return 0;
};
int isKATAKANA(u_code_point s) {
        if (s >= 0x30A0 && s <= 0x30FF) {
            return 1;
        }
        return 0;
};
int isHINDI(u_code_point s) {
```

```
        if (s >= 0x0900 && s <= 0x0970) {
            return 1;
        }
        return 0;
};
int isLatins(u_code_point s) {
        if (s <  0x370) {
            return 1;
        }
        return 0;
};


// Most frequent 888 Hangeul syllables in Korean BizName
#define HG 888
u_code_point hangeul_freq[HG] = {

 0xd55c,0xad6d,0xd559,0xad50,0xb300,0xace0,0xb4f1,0xcd08,
 0xc911,0xb824,0xd654,0xd604,0xc6d0,0xbb38,0xc721,0xbcd1,
 0xc804,0xc790,0xae30,0xacf5,0xc0b0,0xc5c5,0xacc4,0xbb3c,
 0xb958,0xc6b4,0xb3d9,0xcc28,0xc220,0xd56d,0xbd80,0xd68d,
 0xac74,0xc124,0xcee8,0xd305,0xac15,0xc0dd,0xba85,0xc885,
 0xd569,0xc601,0xb18d,0xbb34,0xc5ed,0xc5f0,0xb9f9,0xc120,
 0xc11c,0xc6b8,0xbe44,0xc2dc,0xc2a4,0xd15c,0xd14d,0xd0dd,
 0xc8fc,0xc2dd,0xd3ec,0xce20,0xbc30,0xb2ec,0xc368,0xaf43,
 0xc815,0xbcf4,0xd1b5,0xc2e0,0xc0c1,0xc0ac,0xd68c,0xc138,
 0xc6a9,0xd611,0xcd9c,0xd310,0xc9c4,0xb791,0xb9e4,0xd5d8,
 0xb0b4,0xc154,0xc1fc,0xd551,0xb0a0,0xb110,0xb370,0xc774,
 0xd648,0xb9c8,0xbc14,0xc624,0xc0bf,0xc9d0,0xc2ed,0xc548,
 0xc18c,0xd504,0xd2b8,0xc6e8,0xbbf8,0xb514,0xc5b4,0xc544,
 0xd53c,0xd30c,0xcf54,0xb9ac,0xceec,0xce7c,0xcf00,0xba54,
 0xd22c,0xc740,0xd589,0xce74,0xb4dc,0xadf8,0xb8f9,0xb9b0,
 0xc6d4,0xb79c,0xc5ec,0xc88b,0xace8,0xce90,0xb9bc,0xd578,
 0xac1c,0xbc1c,0xc5d8,0xc9c0,0xae00,0xb85c,0xbc8c,0xc810,
 0xd574,0xd138,0xd0c8,0xd1a0,0xd3f0,0xc678,0xacfc,0xc694,
 0xc778,0xb137,0xb2f7,0xd154,0xb808,0xcf64,0xcef4,0xd4e8,
 0xd130,0xc5d4,0xd14c,0xbc45,0xd06c,0xc13c,0xb2e5,0xd0c0,
 0xc7a5,0xc57d,0xd488,0xc81c,0xc194,0xb8e8,0xc158,0xbc29,
 0xc1a1,0xc77c,0xd074,0xb7fd,0xb355,0xd615,0xd328,0xd3c9,
 0xc0bc,0xc131,0xb0a8,0xbd81,0xac8c,0xc784,0xd50c,0xb77c,
 0xc6cc,0xb7ec,0xc704,0xc628,0xd658,0xacbd,0xcda9,0xbdf0,
 0xc1c4,0xc564,0xc528,0xc640,0xce58,0xb125,0xc5d0,0xc5e0,
 0xd050,0xc54c,0xd2f0,0xc720,0xbe0c,0xc5d1,0xbe14,0xd29c,
 0xbcc0,0xd638,0xbc95,0xb960,0xae08,0xad11,0xcc9c,0xc18d,
 0xc591,0xd65c,0xccad,0xc988,0xc139,0xd734,0xcf5c,0xb354,
 0xd0dc,0xd398,0xb274,0xb9e5,0xbca8,0xcd95,0xc6f0,0xbca0,
 0xb860,0xb2c9,0xad7f,0xc9c1,0xc2f8,0xc820,0xbe5b,0xc758,
 0xbc84,0xc6f9,0xd558,0xac00,0xc744,0xbc31,0xb124,0xd035,
 0xc288,0xc218,0xd37c,0xcee4,0xbba4,0xb2c8,0xb9c1,0xb450,
 0xbbfc,0xb4e0,0xb95c,0xc655,0xd45c,0xc900,0xc584,0xd2f1,
 0xd765,0xd0d1,0xc870,0xbcf5,0xad6c,0xd2b9,0xbaa9,0xb78c,
```

0xbd09,0xd6c4,0xd0b9,0xd038,0xd48d,0xbcc4,0xc554,0xc96c,
0xd070,0xd61c,0xc5b8,0xb798,0xc560,0xbca4,0xcc98,0xd3f4,
0xaddc,0xd6fc,0xbc00,0xc5c4,0xcde8,0xb984,0xcc3d,0xc30d,
0xb2dd,0xd2f8,0xcea0,0xc824,0xc728,0xd0a4,0xc6c5,0xd64d,
0xc2e4,0xc708,0xd30d,0xcc38,0xd5e4,0xb7f4,0xc625,0xad00,
0xb3cc,0xc608,0xd380,0xc62c,0xc2b9,0xc11d,0xb839,0xb9db,
0xc4f0,0xc0e4,0xadf9,0xd5a5,0xd53d,0xb80c,0xd718,0xb9de,
0xcda4,0xbe4c,0xcd94,0xb9cc,0xd1b1,0xb108,0xafbc,0xba38,
0xc6b0,0xc724,0xd329,0xd480,0xc82f,0xc874,0xc8e4,0xce85,
0xb4e4,0xbcf8,0xbc94,0xb825,0xc559,0xaca8,0xcfe0,0xd584,
0xb3c4,0xb098,0xbaa8,0xb2e4,0xc7ac,0xad8c,0xb178,0xbab0,
0xb2e8,0xc9d1,0xccb4,0xc74c,0xb8cc,0xc99d,0xac70,0xae40,
0xb2f9,0xc57c,0xb974,0xbc15,0xc800,0xac80,0xc785,0xb529,
0xb86f,0xcca0,0xbd88,0xbc18,0xbc88,0xc775,0xbd84,0xc791,
0xc0f5,0xb9ad,0xba55,0xac04,0xad70,0xd6a8,0xb2f4,0xb204,
0xcf58,0xd478,0xc0c8,0xd560,0xac10,0xd0c1,0xcfe8,0xc5fc,
0xc5f4,0xac08,0xc545,0xd5c8,0xd544,0xb809,0xd63c,0xb294,
0xb3c5,0xd568,0xcf13,0xc0c9,0xcd0c,0xb4c0,0xb7ed,0xac01,
0xc735,0xb780,0xc2ec,0xba74,0xba3c,0xaca9,0xce68,0xc871,
0xd76c,0xd669,0xd5ec,0xcc44,0xc9c8,0xc789,0xc561,0xb0c9,
0xb840,0xc83c,0xb208,0xd314,0xcc30,0xc801,0xc555,0xacac,
0xd640,0xc8fd,0xc808,0xbe59,0xd540,0xc5bc,0xc2f1,0xb864,
0xadfc,0xd5cc,0xc300,0xc190,0xbe45,0xac1d,0xd0a8,0xcc99,
0xc2ac,0xb09a,0xad74,0xce60,0xc811,0xc2a8,0xc26c,0xb9bd,
0xb85d,0xb784,0xb179,0xace1,0xacb0,0xd2bc,0xd134,0xd0c4,
0xce5c,0xcc45,0xcc2c,0xc6cd,0xc6c0,0xc568,0xc12c,0xb77d,
0xd3b8,0xd32c,0xd150,0xc7a1,0xbe48,0xb9d0,0xb7c9,0xb180,
0xd38c,0xbbf9,0xbaac,0xba40,0xb989,0xb799,0xb144,0xae38,
0xce21,0xc6c3,0xc308,0xc12f,0xc0b4,0xbc0d,0xb978,0xb760,
0xb378,0xb09c,0xd034,0xbc25,0xb9dd,0xb728,0xb2a5,0xb290,
0xd790,0xcd98,0xc637,0xc21c,0xb9e8,0xb9d8,0xb298,0xb150,
0xae09,0xac24,0xd2c0,0xcea1,0xc20d,0xc1e0,0xbcbd,0xbc38,
0xb871,0xb81b,0xb7a8,0xb304,0xd6c8,0xd3ed,0xd0f1,0xcf10,
0xcef5,0xcd5c,0xcd1d,0xc82c,0xc36c,0xc140,0xc0d8,0xbe75,
0xbe60,0xbe10,0xbd95,0xb7f0,0xb7b5,0xb610,0xb3c8,0xb374,
0xb12c,0xb099,0xb044,0xd788,0xd2f4,0xd1a4,0xd0d0,0xc9dc,
0xc58f,0xc2b4,0xc1a5,0xb3d4,0xafc0,0xadc0,0xd508,0xd3fc,
0xd3d0,0xd39c,0xd399,0xd31c,0xd1a8,0xd131,0xce94,0xcd09,
0xccd0,0xcca8,0xcc60,0xcc3e,0xcc29,0xc9f8,0xc9d5,0xc81d,
0xc7a0,0xc644,0xc2b5,0xbc34,0xb9c9,0xb828,0xb2d8,0xb205,
0xae4c,0xd608,0xd31d,0xc90c,0xc88c,0xc73c,0xc5fd,0xc14b,
0xc0f7,0xbc1d,0xba64,0xb561,0xb524,0xb118,0xb0ad,0xb07c,
0xade0,0xac9c,0xac78,0xcfe1,0xcf69,0xcf04,0xc9f1,0xc695,
0xc573,0xc55e,0xc53d,0xc329,0xc290,0xc19c,0xc0ad,0xbb18,
0xb86c,0xb7fc,0xb545,0xb17c,0xaebc,0xae68,0xacf6,0xd799,
0xd761,0xd655,0xd5db,0xd56b,0xd1f4,0xd0b4,0xce78,0xcc0c,
0xc990,0xc63b,0xc61b,0xc384,0xbd99,0xbd90,0xbcfc,0xb8e9,
0xb7a9,0xb69c,0xb5cc,0xb5a1,0xb518,0xb515,0xb451,0xb3fc,
0xb371,0xb358,0xb2ed,0xb188,0xb0e5,0xaf42,0xace4,0xd720,
0xd700,0xd234,0xd1a1,0xcf70,0xcf08,0xce04,0xc9d3,0xc98c,
0xc813,0xc7bc,0xc70c,0xc570,0xc500,0xc3e0,0xc3d8,0xc2f9,

```
 0xc27d,0xc250,0xc22f,0xc058,0xbe68,0xbe54,0xbcbc,0xbabd,
 0xba58,0xba4d,0xb9b4,0xb8f8,0xb460,0xb380,0xb1cc,0xb192,
 0xb140,0xb128,0xb0c5,0xb0a9,0xb05d,0xaf2c,0xae54,0xad34,
 0xac90,0xd575,0xd401,0xd3a8,0xd1b0,0xd0e0,0xcfc4,0xccbc,
 0xcc4c,0xcc1c,0xcbd4,0xc9da,0xc989,0xc717,0xc635,0xc5ff,
 0xc232,0xbafc,0xb8b0,0xb7ad,0xb5bc,0xb530,0xb4dd,0xb465,
 0xb41c,0xb2d0,0xb057,0xb04c,0xad81,0xac13,0xd749,0xd6cc,
 0xd6a1,0xd601,0xd5f4,0xd54c,0xd47c,0xd3ab,0xd384,0xd31f,
 0xd300,0xd15d,0xd140,0xd0ed,0xd0ec,0xcffc,0xcf8c,0xce89,
 0xce84,0xce75,0xce69,0xcd78,0xcd2c,0xcc10,0xc9dd,0xc999,
 0xc8e0,0xc878,0xc7dd,0xc7c1,0xc7ad,0xc7a3,0xc794,0xc641,
 0xc639,0xc610,0xc5b5,0xc58d,0xc575,0xc530,0xc38c,0xc2f6,
 0xc2ef,0xc258,0xc22d,0xc219,0xc0cc,0xc0b6,0xbfcc,0xbf55,
 0xbe7c,0xbe57,0xbdd4,0xbd24,0xbca7,0xbc1f,0xbc1b,0xbbac,
 0xbab8,0xba67,0xb9f7,0xb9d1,0xb9bf,0xb98e,0xb987,0xb86d,
 0xb81d,0xb818,0xb801,0xb730,0xb6f0,0xb6b1,0xb54c,0xb534,
 0xb454,0xb3cb,0xb385,0xb364,0xb2f5,0xb2db,0xb214,0xb18b,
 0xb11d,0xb0c4,0xb0b5,0xaee8,0xae45,0xacfd,0xac71,0xac19,
 0xac11,0xd79d,0xd78c,0xd69f,0xd48b,0xd3a0,0xd301,0xd0e4,
 0xd0d5,0xd03c,0xcf65,0xcf1c,0xcea3,0xcd1b,0xcc64,0xcabd,
 0xc9c7,0xc950,0xc918,0xc8c4,0xc80a,0xc7c8,0xc74d,0xc719,
 0xc6b1,0xc651,0xc619,0xc5e3,0xc580,0xc557,0xc52c,0xc388,
 0xc2fc,0xc19d,0xc178,0xc174,0xc0ec,0xc0d0,0xc068,0xbf08,
 0xbed0,0xbcd5,0xbc40,0xbc2d,0xbbff,0xbbc0,0xbb58,0xbb44,
 0xba5c,0xba4b,0xba39,0xb9f5,0xb9d9,0xb97c,0xb959,0xb93c,
 0xb8e1,0xb819,0xb738,0xb527,0xb51c,0xb458,0xb284,0xb1e8
};


#define HANGUL_REORDER_BASE 0XB000

u_code_point reorder_hangul(u_code_point s) {
        u_code_point i=HANGUL_REORDER_BASE;
        int k=0;
        for(k=0; k<HG; k++,i++) {
           if(s == hangeul_freq[k]) { return i; };
        };
        k=(s - HANGUL_REORDER_BASE);
        if( k>=0 && k<HG) {
           return hangeul_freq[k];
        };
        return s;
}

u_code_point restore_order_hangul(u_code_point z) {
        u_code_point i=HANGUL_REORDER_BASE;
        int k;
        k=(z - HANGUL_REORDER_BASE);
        if( k>=0 && k<HG) {
           return hangeul_freq[k];
        };
```

```
        for(k=0; k<HG; k++,i++) {
           if(z == hangeul_freq[k]) { return i; };
        };
        return z;
}

//Most frequent 4096 SC/TC characters in CJK
#define UH 4096
u_code_point unihan_freq[UH] = {

 0x4f01,0x696d,0x4e1a,0x5de5,0x7a0b,0x96c6,0x5718,0x56e2,
 0x6709,0x9650,0x8cac,0x8d23,0x4efb,0x516c,0x53f8,0x603b,
 0x90e8,0x767c,0x53d1,0x5c55,0x7ad9,0x70b9,0x958b,0x5f00,
 0x79d1,0x6280,0x8853,0x672f,0x54a8,0x8be2,0x5be6,0x5b9e,
 0x901a,0x4fe1,0x606f,0x7cfb,0x7edf,0x7d71,0x7db2,0x8def,
 0x7edc,0x4e2d,0x5fc3,0x7f51,0x56fd,0x570b,0x969b,0x83ef,
 0x96fb,0x7535,0x5b50,0x8111,0x8166,0x6c23,0x6c14,0x5668,
 0x6a5f,0x6c17,0x529b,0x673a,0x68b0,0x8baf,0x8d44,0x8a71,
 0x8bbe,0x8ba1,0x8a2d,0x8a08,0x5099,0x5408,0x52d5,0x52a8,
 0x5236,0x88fd,0x9020,0x4f5c,0x5907,0x8fd0,0x5efa,0x65b0,
 0x7522,0x4ea7,0x7528,0x54c1,0x5382,0x5e94,0x793c,0x98df,
 0x79df,0x8d41,0x5ee0,0x79ae,0x5168,0x7403,0x5c08,0x7523,
 0x773c,0x955c,0x7f8e,0x5bb9,0x6c7d,0x8f66,0x8eca,0x975e,
 0x4ea4,0x6362,0x5bf9,0x5916,0x85cf,0x4e91,0x9655,0x8a9e,
 0x57df,0x540d,0x6ce8,0x518c,0x5e7f,0x64ad,0x5ee3,0x544a,
 0x4e3b,0x6e90,0x50b3,0x4e92,0x806f,0x8054,0x5149,0x6750,
 0x5927,0x5b66,0x5b78,0x5206,0x682a,0x5f0f,0x76df,0x534f,
 0x59d4,0x5458,0x4f1a,0x6703,0x793e,0x7559,0x5354,0x6559,
 0x519c,0x4e13,0x51fa,0x7248,0x6587,0x5316,0x827a,0x85dd,
 0x4f53,0x6210,0x4eba,0x624d,0x65e5,0x672c,0x9577,0x6708,
 0x751f,0x6cd5,0x5f8b,0x5e08,0x5e2b,0x8303,0x533b,0x7597,
 0x6cbb,0x836f,0x4fdd,0x5065,0x5eb7,0x8eab,0x967a,0x96aa,
 0x8d38,0x6613,0x80a1,0x4efd,0x8f6f,0x4ef6,0x8edf,0x9ad4,
 0x5a92,0x8cfc,0x7269,0x6d41,0x65c5,0x904a,0x97f3,0x6a02,
 0x670d,0x52d9,0x52a1,0x5546,0x4e8b,0x7814,0x7a76,0x6240,
 0x9662,0x88dc,0x7fd2,0x73ed,0x5100,0x60c5,0x5831,0x9023,
 0x987e,0x95ee,0x514d,0x8d39,0x53f0,0x6e7e,0x8ba4,0x8bc1,
 0x8c0d,0x7e3d,0x5834,0x573a,0x8fb2,0x89c6,0x8208,0x8cbb,
 0x91d1,0x5c5e,0x5c6c,0x92fc,0x6a21,0x9435,0x7cbe,0x5bc6,
 0x66f8,0x5c4b,0x5167,0x5e97,0x878d,0x904b,0x8f38,0x5ba2,
 0x8b49,0x5238,0x6295,0x8a17,0x9867,0x554f,0x7d9c,0x8ca1,
 0x7d93,0x7ecf,0x7968,0x9280,0x884c,0x92b7,0x7ba1,0x7406,
 0x8cb8,0x6b3e,0x5c0f,0x57fa,0x81ea,0x8aee,0x8a62,0x5275,
 0x5bb6,0x5177,0x767e,0x8ca8,0x74b0,0x5883,0x76d1,0x5229,
 0x7dda,0x5370,0x5237,0x5730,0x651d,0x5f71,0x88dd,0x98fe,
 0x5283,0x805e,0x7b97,0x547d,0x73e0,0x5bf6,0x9418,0x9336,
 0x5357,0x5dde,0x5c71,0x4e1c,0x6cb3,0x6c5f,0x6e56,0x7701,
 0x5317,0x897f,0x4eac,0x5ddd,0x4e0a,0x6d77,0x4e34,0x5e02,
 0x5929,0x6d25,0x8fde,0x6df1,0x5733,0x7586,0x6e29,0x6d59,
 0x82cf,0x7518,0x8083,0x5b89,0x5fbd,0x590f,0x4e0b,0x6728,
 0x6237,0x4f11,0x95f2,0x5b81,0x5e73,0x4e09,0x6b66,0x6c38,
```

0x7389,0x9633,0x8fbd,0x8d35,0x56db,0x53bf,0x5409,0x77f3,
0x592a,0x677e,0x6c99,0x66f2,0x9752,0x6e05,0x9686,0x9646,
0x4e50,0x83b1,0x666f,0x664b,0x9ec4,0x6dee,0x9e64,0x56fa,
0x9ad8,0x6607,0x961c,0x51e4,0x5b9a,0x5fb7,0x4e39,0x957f,
0x660c,0x535a,0x767d,0x963f,0x53e4,0x547c,0x60e0,0x83f1,
0x77e2,0x5d0e,0x6a2a,0x7acb,0x661f,0x6804,0x6238,0x6771,
0x6751,0x6ca2,0x80b2,0x95a2,0x7e4a,0x7dad,0x9060,0x85e4,
0x65ed,0x785d,0x68ee,0x4eca,0x6d0b,0x771f,0x5b9f,0x6e9d,
0x6b21,0x5d8b,0x798f,0x5ca1,0x5bae,0x8a0a,0x8cc7,0x7530,
0x6fa4,0x5bcc,0x58eb,0x6797,0x76f8,0x5171,0x5cf6,0x6e21,
0x702c,0x842c,0x4e16,0x6851,0x6597,0x6a4b,0x7532,0x6d5c,
0x718a,0x623f,0x7b2c,0x79cb,0x8218,0x4e80,0x962a,0x52dd,
0x7247,0x8cc0,0x524d,0x8c4a,0x6803,0x90a6,0x967d,0x6975,
0x4f50,0x5e78,0x5f18,0x8fd1,0x5f8c,0x9234,0x6749,0x7af9,
0x7279,0x6b8a,0x6839,0x88b4,0x8d8a,0x4e38,0x4f4f,0x7d00,
0x5c3e,0x8352,0x9f8d,0x6817,0x592e,0x5e83,0x5fa1,0x7a4d,
0x53cb,0x4ef2,0x80fd,0x5b87,0x83ca,0x5036,0x697d,0x68a8,
0x611b,0x77e5,0x5a9b,0x5948,0x5c90,0x7fa4,0x99ac,0x57fc,
0x9759,0x5343,0x8449,0x9ce5,0x53d6,0x826f,0x6f5f,0x5f62,
0x58f2,0x7d50,0x969c,0x5bb3,0x5199,0x4e57,0x7dcf,0x753b,
0x9580,0x6c96,0x7e04,0x757f,0x9678,0x533a,0x69cb,0x6a29,
0x52b4,0x691c,0x8b72,0x570f,0x5b85,0x8a3c,0x8cc3,0x4fa1,
0x4f9b,0x7d66,0x6bce,0x8b1b,0x6f14,0x9451,0x9031,0x520a,
0x5175,0x5eab,0x5e9c,0x770c,0x75c5,0x8a8d,0x653f,0x515a,
0x73fe,0x7d4c,0x6e08,0x9053,0x7d44,0x52a0,0x56e3,0x8ee2,
0x9f99,0x6cf0,0x4e4c,0x5434,0x5174,0x4f0a,0x5b9c,0x5cb3,
0x5f20,0x6cd7,0x91cd,0x5e86,0x9675,0x7965,0x78f4,0x76f1,
0x7719,0x53e3,0x57ce,0x8363,0x6625,0x6c60,0x6d2a,0x660e,
0x6eaa,0x5d03,0x6743,0x4e49,0x8d21,0x6e2f,0x6d66,0x6811,
0x5e84,0x5f3a,0x9704,0x548c,0x6d6e,0x6c0f,0x73af,0x59da,
0x8c0a,0x9756,0x5609,0x6d4e,0x6f6d,0x9a6c,0x95e8,0x90fd,
0x5bbe,0x5f81,0x539f,0x8c37,0x6cc9,0x5a01,0x95fb,0x6c34,
0x4f59,0x4e61,0x91ce,0x6c82,0x90d1,0x7edb,0x611f,0x6c11,
0x6843,0x5c45,0x6e38,0x5ce1,0x94a2,0x83b2,0x534e,0x965f,
0x9091,0x7a74,0x8fdb,0x6c49,0x5fe0,0x6865,0x68e3,0x5cad,
0x8f89,0x574a,0x8fdc,0x594e,0x82cd,0x8f7d,0x5e90,0x67f1,
0x6f58,0x6ecb,0x8305,0x90a1,0x5830,0x676d,0x953a,0x7a37,
0x9976,0x865e,0x9634,0x8fbe,0x768b,0x5bff,0x6000,0x82d1,
0x971e,0x679c,0x5ea6,0x51c9,0x9065,0x9526,0x666e,0x6ce2,
0x96c4,0x76ae,0x5145,0x4faf,0x4e30,0x5be8,0x5e95,0x5ca9,
0x4e95,0x74a7,0x6cad,0x7317,0x6cfd,0x9896,0x6c7e,0x8821,
0x829d,0x829c,0x9c81,0x5c01,0x8d24,0x5854,0x575b,0x5802,
0x6cb9,0x74ef,0x5cea,0x58a8,0x9a85,0x6885,0x5188,0x4ec1,
0x57a3,0x58c1,0x80a5,0x95f4,0x90f8,0x4f26,0x62c9,0x5c14,
0x59cb,0x853a,0x4e08,0x6d6a,0x6df3,0x6986,0x5510,0x7b60,
0x8981,0x90ae,0x88d5,0x987a,0x9f0e,0x6c9f,0x51f0,0x79ba,
0x65bd,0x6566,0x714c,0x5300,0x8425,0x839e,0x5934,0x80dc,
0x8fb9,0x5f92,0x8354,0x719f,0x7f57,0x9e21,0x4ead,0x57e0,
0x94f6,0x5f66,0x5df4,0x6556,0x56fe,0x52d2,0x575d,0x978d,
0x9738,0x67cf,0x868c,0x5305,0x5b9d,0x6ee8,0x52c3,0x6cca,
0x66f9,0x8336,0x5e38,0x671d,0x6f6e,0x5de2,0x90f4,0x6f84,

0x627f,0x8d64,0x5d07,0x6ec1,0x695a,0x6148,0x4ece,0x5355,
0x5f53,0x7a3b,0x767b,0x9093,0x8fea,0x6d1e,0x5ce8,0x5d4b,
0x5a25,0x9102,0x6069,0x756a,0x65b9,0x9632,0x5949,0x4f5b,
0x6276,0x629a,0x683c,0x4e2a,0x5de9,0x6842,0x54c8,0x90af,
0x542b,0x8377,0x83cf,0x8d3a,0x9ed1,0x5b88,0x8861,0x7ea2,
0x846b,0x82a6,0x864e,0x82b1,0x6ed1,0x69d0,0x83b7,0x970d,
0x7ee9,0x5373,0x5180,0x5939,0x4f73,0x7b80,0x5251,0x59dc,
0x5c06,0x7126,0x80f6,0x63ed,0x4ecb,0x8346,0x4e5d,0x9152,
0x53e5,0x5580,0x51ef,0x514b,0x57a6,0x5e93,0x6606,0x5170,
0x5eca,0x8001,0x96f7,0x51b7,0x4e3d,0x5ec9,0x6d9f,0x6881,
0x804a,0x7075,0x67f3,0x516d,0x5a04,0x9e7f,0x6f5e,0x6ee6,
0x6d1b,0x6ee1,0x8302,0x7709,0x8499,0x5b5f,0x7c73,0x7ef5,
0x95fd,0x7261,0x7a46,0x5ae9,0x76d8,0x84ec,0x5f6d,0x6c9b,
0x78d0,0x840d,0x8386,0x84b2,0x6816,0x4e03,0x9f50,0x7941,
0x542f,0x8fc1,0x6f5c,0x94a6,0x743c,0x90b1,0x5982,0x4e73,
0x6c5d,0x745e,0x838e,0x8272,0x6c55,0x5c1a,0x91b4,0x9edf,
0x97f6,0x5173,0x90b5,0x7ecd,0x5c04,0x5341,0x4ec0,0x8212,
0x53cc,0x6714,0x601d,0x5bbf,0x968f,0x7ee5,0x9042,0x68e0,
0x94c1,0x6850,0x540c,0x94dc,0x4e07,0x6c6a,0x65fa,0x671b,
0x5fae,0x6f4d,0x6e2d,0x536b,0x74ee,0x6da1,0x65e0,0x68a7,
0x4e94,0x821e,0x9521,0x53a6,0x4ed9,0x54b8,0x732e,0x9999,
0x8944,0x6e58,0x54cd,0x9879,0x8c61,0x8427,0x5b5d,0x8f9b,
0x5ffb,0x90a2,0x5f90,0x4fee,0x53d9,0x8bb8,0x859b,0x65ec,
0x5bfb,0x96c5,0x70df,0x76d0,0x5ef6,0x6cbf,0x626c,0x4eea,
0x76ca,0x82f1,0x9e70,0x79b9,0x5143,0x8d5e,0x67a3,0x589e,
0x624e,0x5c6f,0x6cbe,0x6e5b,0x6a1f,0x7ae0,0x6f33,0x62db,
0x662d,0x8d75,0x8087,0x9547,0x6b63,0x679d,0x821f,0x5468,
0x8bf8,0x9a7b,0x6dc4,0x7d2b,0x90b9,0x9075,0x5de6,0x5043,
0x510b,0x5156,0x4eb3,0x9097,0x90b3,0x90d3,0x90eb,0x90ef,
0x5152,0x7ae5,0x8297,0x82ae,0x8392,0x834f,0x8365,0x8398,
0x8572,0x5c91,0x5c9a,0x5d4a,0x5d69,0x8862,0x9606,0x6c76,
0x6cf8,0x6cfe,0x6d4f,0x6d60,0x6dc7,0x6dc5,0x6dbf,0x6e11,
0x6e5f,0x6e44,0x6ea7,0x6f62,0x6fa7,0x6fee,0x7f19,0x9095,
0x73f2,0x679e,0x67d8,0x6866,0x683e,0x6ed5,0x65cc,0x7800,
0x7684,0x662f,0x4e00,0x4e0d,0x6211,0x4e86,0x5728,0x5230,
0x4ed6,0x4f60,0x4ee5,0x53ef,0x5c31,0x4e5f,0x597d,0x8fd9,
0x90a3,0x5f97,0x0000,0x6765,0x4e4b,0x5e74,0x53bb,0x591a,
0x770b,0x9019,0x500b,0x800c,0x60f3,0x8bf4,0x4eec,0x70ba,
0x53ea,0x4f86,0x7136,0x4e3a,0x63d0,0x5979,0x65f6,0x6642,
0x4f46,0x5f88,0x8aaa,0x6c92,0x8d77,0x624b,0x610f,0x53c8,
0x4e9b,0x904e,0x5176,0x9762,0x8acb,0x7740,0x5011,0x6b64,
0x6700,0x8fc7,0x91cc,0x5df2,0x4f55,0x56e0,0x9ebc,0x8005,
0x4e8c,0x540e,0x4f4d,0x9084,0x5c0d,0x5973,0x4e48,0x5df1,
0x56de,0x628a,0x518d,0x6253,0x6bd4,0x6ca1,0x4f7f,0x4e8e,
0x88ab,0x7b49,0x8fd8,0x5c11,0x6216,0x7121,0x65bc,0x6027,
0x5427,0x7576,0x5411,0x55ce,0x5148,0x5404,0x7531,0x5165,
0x89c1,0x53ca,0x4fbf,0x505a,0x50cf,0x671f,0x4ee3,0x76ee,
0x89e3,0x9ede,0x984c,0x8868,0x5462,0x8d70,0x4e24,0x66f4,
0x6a23,0x81f3,0x6837,0x73b0,0x5b83,0x6d3b,0x4e0e,0x600e,
0x795e,0x653e,0x6821,0x8b1d,0x8457,0x5feb,0x63a5,0x6b7b,
0x53cd,0x8207,0x738b,0x5b57,0x53d7,0x79cd,0x58f0,0x7b11,

0x627e,0x76f4,0x53eb,0x8bdd,0x513f,0x6bcf,0x8a00,0x61c9,
0x7a2e,0x5b8c,0x6307,0x51e0,0x7ed9,0x529f,0x559c,0x82e5,
0x5f1f,0x8ddf,0x95dc,0x754c,0x60a8,0x9593,0x9cf3,0x5fc5,
0x89ba,0x8a72,0x6539,0x5426,0x516b,0x7a7a,0x554a,0x9032,
0x5566,0x5403,0x4e14,0x5c07,0x5169,0x7537,0x8003,0x6c42,
0x542c,0x5e76,0x8ad6,0x5185,0x672a,0x5225,0x807d,0x6301,
0x5019,0x898b,0x88e1,0x98a8,0x5374,0x519b,0x7063,0x91cf,
0x534a,0x5e0c,0x5f80,0x522b,0x5904,0x674e,0x73a9,0x66fe,
0x5931,0x5340,0x4e66,0x932f,0x5b69,0x54ea,0x6536,0x8b93,
0x62ff,0x4ee4,0x9078,0x62a5,0x8f03,0x751a,0x6578,0x652f,
0x5f9e,0x4f3c,0x6b61,0x96be,0x6570,0x6bdb,0x6b65,0x65e9,
0x822c,0x5e7e,0x706b,0x9700,0x53e6,0x592b,0x4e4e,0x96e3,
0x982d,0x5ba4,0x6599,0x5012,0x8a31,0x4eb2,0x6574,0x5e72,
0x8cb7,0x8a18,0x5144,0x865f,0x670b,0x843d,0x8655,0x9996,
0x65af,0x9664,0x6bb5,0x6015,0x5ff5,0x6545,0x793a,0x63a8,
0x4e45,0x5947,0x4e26,0x7236,0x5f35,0x665a,0x5207,0x8bb0,
0x7834,0x53f2,0x5fd7,0x8ab0,0x98ce,0x7167,0x6218,0x7adf,
0x5f15,0x54e5,0x89c9,0x9898,0x5f85,0x6848,0x8bf7,0x5b58,
0x7231,0x8ba9,0x5c40,0x591c,0x82e6,0x7b54,0x901f,0x6b4c,
0x9673,0x8bba,0x8f49,0x9ee8,0x6d3e,0x5361,0x8b8a,0x8a66,
0x6d88,0x7ed3,0x602a,0x8db3,0x677f,0x5dee,0x55ae,0x7fa9,
0x5217,0x578b,0x9769,0x6230,0x961f,0x5750,0x968a,0x537b,
0x6392,0x5e26,0x8d85,0x5047,0x9001,0x5beb,0x5b98,0x6761,
0x8072,0x53d8,0x8be5,0x81fa,0x9886,0x4f20,0x6bcd,0x54e1,
0x6389,0x8a0e,0x67e5,0x5247,0x51b3,0x6a94,0x5475,0x4f4e,
0x4ecd,0x59b3,0x529e,0x521d,0x5e03,0x5f37,0x8b70,0x52a9,
0x8fa6,0x50f9,0x571f,0x8f6c,0x505c,0x4f17,0x8f7b,0x5ea7,
0x503c,0x6562,0x8bed,0x65cf,0x8ff7,0x7a81,0x53f3,0x6c7a,
0x67d0,0x8bc6,0x6781,0x7d1a,0x8840,0x8036,0x820d,0x8138,
0x8dd1,0x94b1,0x523b,0x6025,0x4f9d,0x5594,0x6551,0x6a19,
0x7368,0x5386,0x89d2,0x5fd8,0x8c93,0x6548,0x75db,0x9ec3,
0x53c3,0x4f8b,0x8bae,0x8996,0x89c0,0x51c6,0x8863,0x9645,
0x5219,0x6279,0x636e,0x6162,0x5bfc,0x638c,0x9322,0x5531,
0x5fd9,0x80cc,0x6982,0x5473,0x5200,0x7591,0x9304,0x8bfb,
0x98de,0x89c2,0x4e89,0x5e1d,0x63db,0x7ec4,0x81f4,0x6309,
0x79bb,0x867d,0x6b62,0x786c,0x7f16,0x5e6b,0x78ba,0x8c08,
0x8ffd,0x7387,0x5c3d,0x8bb2,0x985e,0x6740,0x756b,0x8c03,
0x8a34,0x9047,0x6fc0,0x559d,0x65e2,0x5e36,0x667a,0x9644,
0x6697,0x7ec8,0x65c1,0x80e1,0x59b9,0x59d0,0x8da3,0x7ea7,
0x5716,0x68d2,0x7bc7,0x8cfd,0x7761,0x8b58,0x908a,0x914d,
0x6bd2,0x96e8,0x51b2,0x96d6,0x4eae,0x6b0a,0x5584,0x9a57,
0x4e3e,0x6293,0x5a18,0x8349,0x8b80,0x8df3,0x98db,0x561b,
0x5440,0x70ed,0x6eff,0x5922,0x5ba3,0x8ab2,0x8ecd,0x79f0,
0x7f6a,0x7d04,0x7a7f,0x7ea6,0x9858,0x60ca,0x5417,0x9000,
0x653b,0x9054,0x53f7,0x90ed,0x7edd,0x9009,0x7d20,0x53c2,
0x8b66,0x4e9a,0x590d,0x4f24,0x7c7b,0x5e2d,0x5bc4,0x6b22,
0x725b,0x52bf,0x65ad,0x9648,0x61c2,0x5920,0x5348,0x4ef7,
0x5224,0x789f,0x59d3,0x62b1,0x8ac7,0x8ce3,0x89c4,0x5988,
0x521a,0x663e,0x5b97,0x6e96,0x6c89,0x5747,0x8089,0x613f,
0x6cc1,0x786e,0x724c,0x96e2,0x6388,0x4ea6,0x5c0e,0x72d7,
0x7d27,0x5e2e,0x4f2f,0x7ebf,0x9760,0x5a5a,0x8abf,0x526f,

0x6768,0x8857,0x50b7,0x525b,0x541b,0x8282,0x83ab,0x5957,
0x5509,0x88c5,0x7f6e,0x54b1,0x6bba,0x5ffd,0x5c81,0x6563,
0x7b56,0x689d,0x60b2,0x4e25,0x72c2,0x7d55,0x62dc,0x7a31,
0x7eaa,0x64da,0x811a,0x76e1,0x9732,0x6807,0x70c8,0x5712,
0x5c3c,0x996d,0x6050,0x641e,0x59d1,0x72af,0x5bdf,0x8ff0,
0x96d9,0x63a7,0x51b5,0x7d05,0x6b32,0x51fb,0x55ef,0x4ec5,
0x7a97,0x5a46,0x5347,0x6838,0x77ed,0x7eed,0x7687,0x57f7,
0x7565,0x72ec,0x66b4,0x67b6,0x4e70,0x62a4,0x9b54,0x96f2,
0x7aef,0x7f3a,0x91c7,0x7956,0x9808,0x5fcd,0x6d32,0x9b3c,
0x8cea,0x80af,0x8077,0x4e71,0x62cd,0x5fa9,0x96ea,0x5218,
0x7bc0,0x898f,0x7562,0x5f04,0x71b1,0x9ebb,0x9928,0x7237,
0x5212,0x6297,0x614b,0x4ed8,0x552e,0x89aa,0x4e82,0x5f69,
0x62ec,0x5634,0x5178,0x9519,0x521b,0x64ca,0x8209,0x987b,
0x7d42,0x7533,0x79fb,0x8239,0x6458,0x65b7,0x8f15,0x7c21,
0x97ff,0x96a8,0x7df4,0x5e55,0x7e8c,0x9b5a,0x54ed,0x804c,
0x7ec6,0x8bc9,0x6001,0x79c1,0x964d,0x7b14,0x656c,0x5757,
0x77a7,0x79c0,0x60dc,0x5e79,0x9910,0x5c0a,0x5de8,0x8d28,
0x7f85,0x7981,0x9ed8,0x5438,0x907f,0x97e6,0x56f0,0x56f4,
0x83dc,0x5446,0x56ed,0x6731,0x6a13,0x54c7,0x501f,0x7169,
0x591f,0x8d5b,0x6f2b,0x4fca,0x986f,0x8f83,0x78bc,0x9192,
0x675f,0x5c24,0x697c,0x5b59,0x7238,0x7d22,0x523a,0x5077,
0x552f,0x8bd7,0x8056,0x5cf0,0x58de,0x704c,0x654c,0x8bd5,
0x9810,0x8c22,0x51e1,0x773e,0x504f,0x4f38,0x722d,0x9a8c,
0x8aa4,0x6df7,0x5ead,0x5806,0x9806,0x8033,0x9aa8,0x517b,
0x8cb4,0x900f,0x8ca0,0x58d3,0x6076,0x9069,0x4eab,0x4fc2,
0x7ef4,0x51b0,0x6e2c,0x6e10,0x61f7,0x5de7,0x8fce,0x5360,
0x79d8,0x5f02,0x6d17,0x55da,0x8d1f,0x4ea1,0x8a55,0x9635,
0x5c42,0x7d30,0x5e8f,0x9003,0x5b63,0x4f19,0x91ab,0x7ec7,
0x9986,0x904d,0x5e8a,0x7434,0x4e60,0x775b,0x7763,0x6200,
0x5f52,0x4e01,0x63f4,0x67d4,0x6557,0x4e1d,0x5371,0x7a3f,
0x694a,0x5740,0x51a0,0x723d,0x6b23,0x62bd,0x52b3,0x684c,
0x59bb,0x5987,0x6298,0x9748,0x52c7,0x6068,0x9a0e,0x4ed4,
0x8bc4,0x9014,0x9805,0x6232,0x63a2,0x5565,0x7686,0x5fb5,
0x6311,0x6beb,0x8c6a,0x52aa,0x672b,0x6258,0x53f6,0x72d0,
0x86cb,0x6628,0x538b,0x71df,0x594f,0x66ff,0x5956,0x8d76,
0x6b77,0x723e,0x5f55,0x7de8,0x9707,0x5954,0x8b77,0x9f13,
0x987f,0x64cd,0x5b64,0x64c7,0x4ebf,0x6167,0x7ee7,0x7d2f,
0x6b72,0x654f,0x4f34,0x805a,0x96bb,0x4f18,0x9669,0x9818,
0x9636,0x62c5,0x63d2,0x5c0b,0x949f,0x8bbf,0x5377,0x6eab,
0x990a,0x8ba8,0x5bd2,0x6447,0x5999,0x6784,0x7ec3,0x5f31,
0x8b02,0x7570,0x906d,0x512a,0x8feb,0x6325,0x721b,0x78b0,
0x5fcc,0x63e1,0x5976,0x9694,0x60e1,0x7eb8,0x6108,0x9876,
0x72c0,0x4e58,0x5439,0x5356,0x6478,0x5433,0x795d,0x68a6,
0x8a5e,0x5287,0x96f6,0x5267,0x563f,0x817f,0x90ce,0x975c,
0x575a,0x6f02,0x5e7b,0x731c,0x73cd,0x4e9e,0x7259,0x6742,
0x5cb8,0x9010,0x9663,0x65e7,0x56b4,0x5076,0x58d8,0x4e43,
0x539a,0x52e2,0x80f8,0x79ef,0x7239,0x76db,0x7f62,0x9022,
0x862d,0x7de3,0x7c3d,0x4e88,0x558a,0x822a,0x8131,0x5f39,
0x563b,0x7ffb,0x574f,0x883b,0x5f7c,0x9c9c,0x5708,0x6bd5,
0x6234,0x5192,0x7d61,0x6469,0x54f2,0x8f2f,0x4e7e,0x65d7,
0x6b27,0x8d99,0x6790,0x5c9b,0x820a,0x68cb,0x96dc,0x8d25,

0x67aa,0x9002,0x9e97,0x865a,0x9884,0x7bb1,0x7eb7,0x9500,
0x78c1,0x9c7c,0x7206,0x7c4d,0x8173,0x528d,0x5b8b,0x6b49,
0x6241,0x5b8f,0x706f,0x72b6,0x616e,0x7d39,0x5289,0x888b,
0x8ba2,0x61b6,0x8af8,0x7b26,0x9a82,0x8f93,0x632f,0x731b,
0x8bcd,0x53ec,0x7f75,0x7d14,0x6cea,0x4fd7,0x8aa0,0x8d22,
0x7e7c,0x54e6,0x6620,0x7cca,0x585e,0x91cb,0x8ddd,0x51ac,
0x7a0d,0x74f6,0x649e,0x84c9,0x9375,0x8d95,0x780d,0x7b46,
0x8a3b,0x5269,0x71d5,0x6028,0x7f8a,0x6a39,0x500d,0x69ae,
0x5ba1,0x5899,0x5723,0x8dc3,0x966a,0x6b78,0x6267,0x5bc2,
0x6653,0x5f48,0x57f9,0x885b,0x7070,0x4e56,0x9298,0x72fc,
0x8f88,0x584a,0x5c16,0x95ea,0x9690,0x52b2,0x6f22,0x95f9,
0x5385,0x67d3,0x8521,0x8cde,0x8f09,0x6101,0x7eff,0x62d6,
0x5766,0x4f0d,0x6c88,0x6094,0x82b3,0x6155,0x989d,0x56c9,
0x8bef,0x87a2,0x8010,0x5049,0x85a6,0x7eb3,0x8c46,0x6c61,
0x555f,0x80a9,0x62b5,0x9057,0x71c8,0x6aa2,0x65e6,0x5abd,
0x633a,0x8d27,0x8e0f,0x50bb,0x62d4,0x4ec7,0x7f13,0x8c6c,
0x4ef0,0x4f1f,0x6fdf,0x8881,0x642d,0x8a13,0x70e7,0x85cd,
0x7b28,0x6668,0x8170,0x80d6,0x62a2,0x64d4,0x88c1,0x7d19,
0x8cbc,0x620f,0x8fc5,0x6ce1,0x642c,0x8c13,0x7f77,0x8bfe,
0x8a73,0x517c,0x5b54,0x6084,0x963b,0x53d4,0x81c2,0x903c,
0x9b42,0x62e5,0x81c9,0x788e,0x53f9,0x63cf,0x4f69,0x7e41,
0x62d2,0x6302,0x54c0,0x734e,0x6ce5,0x70ae,0x7b7e,0x6575,
0x9109,0x518a,0x8f2a,0x62ac,0x8f6e,0x8bad,0x5706,0x5c3a,
0x885d,0x622a,0x91ca,0x593a,0x9df9,0x6d4b,0x76d6,0x68c4,
0x9605,0x8d2d,0x78e8,0x8000,0x5e45,0x9189,0x7e23,0x7dca,
0x7eb5,0x62e9,0x8c8c,0x50c5,0x5e33,0x5c64,0x9f20,0x9677,
0x93e1,0x5435,0x6089,0x4fc3,0x62fc,0x54e9,0x8a89,0x8d0a,
0x8986,0x978b,0x68c0,0x5bab,0x6c57,0x59ca,0x7897,0x8eb2,
0x9846,0x65cb,0x5410,0x5e7d,0x74dc,0x6de1,0x4fb5,0x9f3b,
0x8a69,0x66c9,0x6446,0x60d1,0x5965,0x6d89,0x5e3d,0x4eff,
0x64c1,0x706d,0x6176,0x7e3e,0x660f,0x8651,0x7345,0x5bbd,
0x570d,0x9918,0x5761,0x50e7,0x9b25,0x8865,0x70b8,0x7bc4,
0x8f1d,0x8b6f,0x5c41,0x72e0,0x6212,0x5ef3,0x7a33,0x722c,
0x8896,0x6c47,0x84cb,0x5211,0x7c97,0x5389,0x5c4a,0x516e,
0x8584,0x63ee,0x8ff9,0x6770,0x76fe,0x9178,0x6735,0x606d,
0x9a5a,0x78a9,0x8dcc,0x7c43,0x4e1f,0x76e4,0x6b3a,0x4e4f,
0x6355,0x6070,0x5fc6,0x54a7,0x5bfa,0x5e25,0x9080,0x8bda,
0x51cc,0x51cf,0x7384,0x865b,0x907a,0x4f0f,0x639b,0x9ebd,
0x9488,0x7ade,0x6717,0x9177,0x7c89,0x6ec5,0x609f,0x809a,
0x6691,0x8bfa,0x6b8b,0x8a8c,0x5713,0x54ac,0x5272,0x707e,
0x90aa,0x77db,0x98ef,0x4e54,0x75be,0x5a03,0x5e7c,0x7cae,
0x9802,0x8bd1,0x4fe0,0x8c0b,0x7840,0x4fc4,0x635f,0x96de,
0x8f86,0x501a,0x51c0,0x8afe,0x8f14,0x5f79,0x76c8,0x675c,
0x7bad,0x81e8,0x7f72,0x4f30,0x6170,0x80de,0x5538,0x63aa,
0x6190,0x8607,0x6e1b,0x81ed,0x51dd,0x8361,0x76fc,0x760b,
0x88c2,0x6643,0x83f2,0x594b,0x82ac,0x80c6,0x5f03,0x70e6,
0x63a1,0x5eb8,0x5c46,0x72b9,0x7a0e,0x8f29,0x85e5,0x9a19,
0x7da0,0x7e2e,0x7372,0x950b,0x62c6,0x6696,0x586b,0x50b2,
0x7262,0x60ef,0x6492,0x59c6,0x51ed,0x5e01,0x52e4,0x59a8,
0x6f38,0x659c,0x7801,0x8106,0x5ee2,0x6dda,0x6a1e,0x626f,
0x4e32,0x7a77,0x9887,0x8d0f,0x6b50,0x503e,0x5306,0x8a02,

0x97e9,0x7720,0x5587,0x98d8,0x8fb1,0x6263,0x89f8,0x8ce2,
0x79e6,0x5091,0x4fa7,0x812b,0x86c7,0x8d4f,0x7cdf,0x845b,
0x4f48,0x690d,0x6062,0x7eaf,0x95ed,0x8eba,0x62b9,0x60a0,
0x5141,0x626b,0x74e6,0x81e3,0x541f,0x84bc,0x53ad,0x6ef4,
0x5df7,0x5805,0x8d34,0x78a7,0x64e6,0x6377,0x61f6,0x6eda,
0x8e22,0x7f18,0x751c,0x8d1d,0x6da6,0x6251,0x8fd4,0x6905,
0x6d69,0x7a69,0x6269,0x73b2,0x680f,0x7272,0x5413,0x4fe9,
0x84dd,0x7d72,0x5875,0x6163,0x6fe4,0x8abc,0x4f54,0x9a91,
0x6350,0x5c60,0x626d,0x8cf4,0x968e,0x62fe,0x8c50,0x989c,
0x8d2b,0x8c9d,0x5018,0x90f5,0x85c9,0x6cdb,0x58ee,0x8428,
0x4ff1,0x5978,0x96b1,0x75bc,0x5fe7,0x9ece,0x8150,0x6158,
0x6454,0x7f9e,0x832b,0x9b4f,0x7d0d,0x827e,0x5bde,0x72f1,
0x89e6,0x6930,0x582a,0x65a4,0x5c48,0x604b,0x912d,0x5acc,
0x59ff,0x7159,0x502b,0x57cb,0x6416,0x5893,0x8d6b,0x901d,
0x5c82,0x75f4,0x699c,0x7e54,0x8058,0x676f,0x6e9c,0x6349,
0x4fa0,0x7ffc,0x8fdf,0x6f0f,0x6316,0x6676,0x60a3,0x7f29,
0x51f6,0x8f9e,0x9f84,0x8907,0x5f84,0x5de1,0x8d56,0x838a,
0x4e59,0x66f0,0x6124,0x5b99,0x60e8,0x63a9,0x82d7,0x5bf8,
0x9ea6,0x83e9,0x64fe,0x63da,0x5782,0x817e,0x9038,0x55b5,
0x54fc,0x7b51,0x7661,0x66fc,0x983b,0x67ef,0x5c97,0x7fc1,
0x94fa,0x91dd,0x71c3,0x6321,0x8de8,0x66ab,0x6d82,0x886b,
0x9670,0x5ef7,0x4ed7,0x6bb7,0x526a,0x5e10,0x8fa8,0x67f4,
0x7a00,0x6f20,0x52ff,0x732a,0x5915,0x7626,0x8179,0x8d74,
0x8fa3,0x529d,0x7b4b,0x87f2,0x71d2,0x6572,0x75c7,0x5112,
0x9801,0x8a93,0x79d2,0x52ab,0x6324,0x856d,0x543e,0x6590,
0x6d01,0x5be7,0x51fd,0x61b2,0x708e,0x5974,0x64a5,0x9e23,
0x5ac1,0x9676,0x5c38,0x626e,0x9aee,0x6fb3,0x5f6c,0x6cf3,
0x9897,0x9f9c,0x7fbd,0x5f6a,0x8389,0x984f,0x64a4,0x9592,
0x4e27,0x6b98,0x7267,0x5582,0x76d2,0x8205,0x61be,0x8017,
0x57c3,0x540a,0x6247,0x6296,0x70c2,0x9d3b,0x871c,0x9875,
0x96d5,0x8faf,0x796d,0x64ec,0x9055,0x5c18,0x6bbf,0x6182,
0x68af,0x996e,0x6d3d,0x5c4f,0x4f8d,0x52de,0x5be2,0x7fe0,
0x65e8,0x7eea,0x6daf,0x52c9,0x6d8c,0x6236,0x7fd4,0x7433,
0x984d,0x8d3c,0x64fa,0x9006,0x6a6b,0x53db,0x6127,0x5e9f,
0x556a,0x8be6,0x6c64,0x5f7b,0x758f,0x8f70,0x6328,0x9f4a,
0x601c,0x5f26,0x68da,0x8cfa,0x6efe,0x62ab,0x9e1f,0x85aa,
0x806a,0x8fa9,0x6bc1,0x8b5c,0x866b,0x997f,0x6109,0x70cf,
0x77ad,0x5339,0x67ab,0x9b06,0x6c1b,0x97ad,0x9640,0x6323,
0x75b2,0x5783,0x5a1c,0x5f2f,0x80ce,0x5687,0x6db2,0x87f9,
0x9b27,0x573e,0x52fe,0x848b,0x5203,0x75d5,0x5b6b,0x7199,
0x8fdd,0x4e8f,0x6b20,0x7260,0x641c,0x59a5,0x820c,0x4e22,
0x9396,0x51f1,0x640d,0x67c4,0x5951,0x7ed5,0x4e10,0x679a,
0x6dfb,0x4e11,0x6682,0x8070,0x73ab,0x614c,0x9012,0x6dd1,
0x8ff4,0x7af6,0x8702,0x60f9,0x5448,0x53b2,0x9e3f,0x715e,
0x6de8,0x901b,0x727d,0x621a,0x888d,0x95f7,0x5496,0x611a,
0x6e34,0x52b1,0x8230,0x5f70,0x5085,0x7275,0x7483,0x98c4,
0x4e18,0x7235,0x6367,0x6021,0x6dfa,0x59fb,0x7802,0x5851,
0x65a5,0x737b,0x75af,0x9eb5,0x541e,0x8266,0x5821,0x607c,
0x7e31,0x6016,0x9583,0x98a4,0x84ee,0x73bb,0x9ea5,0x6bc5,
0x95b1,0x8ad2,0x7cd6,0x5351,0x52a3,0x5be9,0x6bc0,0x6674,
0x53ed,0x6291,0x8270,0x7470,0x95c6,0x73ca,0x6191,0x94bb,

0x5561,0x93ae,0x8870,0x5ed6,0x90c1,0x8877,0x6168,0x50ac,
0x732b,0x7a79,0x6d9b,0x5146,0x92d2,0x7cd5,0x5bec,0x64ce,
0x6602,0x9505,0x62f3,0x7891,0x614e,0x9a45,0x5353,0x7f5a,
0x76c6,0x6d53,0x952e,0x8109,0x90bb,0x9501,0x9817,0x9063,
0x59ae,0x81e5,0x6b6a,0x5507,0x524a,0x9a7e,0x7978,0x9059,
0x880d,0x5967,0x9f4b,0x51a4,0x69cd,0x8096,0x89c8,0x9589,
0x62d3,0x5751,0x9813,0x810f,0x77ee,0x8180,0x6863,0x52f5,
0x629b,0x8679,0x9a71,0x7a9d,0x88e4,0x543b,0x9614,0x6dcb,
0x8a2a,0x655d,0x739b,0x9891,0x7985,0x7f69,0x85a9,0x98f2,
0x95a3,0x7838,0x5c1d,0x4ea8,0x7c92,0x576a,0x68cd,0x76d7,
0x76f2,0x5deb,0x7b79,0x964c,0x6436,0x5be1,0x77ac,0x6d45,
0x5154,0x53c9,0x778e,0x7dd2,0x77e9,0x4e0c,0x5490,0x5118,
0x6383,0x62bc,0x6b04,0x5617,0x6b96,0x538c,0x52f8,0x72c4,
0x7ff0,0x8d81,0x800d,0x8dea,0x9bae,0x7092,0x596e,0x8cdc,
0x5764,0x95e1,0x9274,0x8d8b,0x64f4,0x6ce3,0x742a,0x54aa,
0x8d2f,0x5d14,0x62e8,0x900a,0x8154,0x76dc,0x8482,0x8d54,
0x7f70,0x8c6b,0x67af,0x6495,0x7cb9,0x846c,0x68c9,0x88ad,
0x54ce,0x9a76,0x60e7,0x7ebd,0x54c9,0x76e3,0x8e2a,0x7c4c,
0x4e1b,0x9072,0x9510,0x8a87,0x8776,0x7a05,0x6e7f,0x7741,
0x77e3,0x7f50,0x7d1b,0x6746,0x6d51,0x8d62,0x5a36,0x9a70,
0x6052,0x70e4,0x8a95,0x9b31,0x7832,0x5996,0x7246,0x9970,
0x7f38,0x7aa9,0x507f,0x50be,0x7f20,0x8fad,0x6756,0x6d74,
0x62d8,0x6254,0x6444,0x6876,0x62df,0x6208,0x8f1b,0x6d12,
0x9f61,0x95ef,0x7b52,0x5026,0x8fb0,0x745c,0x716e,0x813e,
0x9971,0x7f1d,0x908f,0x8c48,0x6dbc,0x6b47,0x7378,0x79e9,
0x5f17,0x54a6,0x84c4,0x5f91,0x70bc,0x6f5b,0x5baa,0x5e99,
0x8292,0x8155,0x50a8,0x535c,0x51af,0x5524,0x7336,0x8d2a,
0x906e,0x6270,0x5367,0x7bc9,0x53ee,0x8f9c,0x5442,0x80c1,
0x5bb0,0x5a49,0x7fc5,0x5674,0x6591,0x68f5,0x5a66,0x8c31,
0x5e63,0x638f,0x5984,0x58ef,0x8e29,0x99a8,0x6deb,0x9601,
0x532a,0x6614,0x7164,0x9e4f,0x9b6f,0x5104,0x59e5,0x5362,
0x82af,0x54bd,0x6065,0x7efc,0x7b1b,0x5352,0x6368,0x9b45,
0x7779,0x6655,0x633d,0x8247,0x62e6,0x6e6f,0x6014,0x7aae,
0x52c1,0x745f,0x6b67,0x67dc,0x522e,0x77aa,0x6f06,0x81bd,
0x96fe,0x919c,0x5c3f,0x8e8d,0x7e73,0x7f55,0x5319,0x5bb4,
0x803b,0x8086,0x644a,0x5835,0x97d3,0x5f65,0x99d5,0x8822,
0x54b3,0x7de9,0x6012,0x8bde,0x6846,0x60ac,0x634f,0x7334,
0x537f,0x71ac,0x6046,0x4f51,0x6789,0x552c,0x51d1,0x80c3,
0x5e15,0x964b,0x55bb,0x8ed2,0x5492,0x5589,0x60f6,0x5a9a,
0x8299,0x541d,0x7b3c,0x98a0,0x5f4e,0x5288,0x643a,0x5537,
0x8ce6,0x6cc4,0x809d,0x754f,0x63b7,0x5429,0x522a,0x7ea0,
0x66ae,0x7919,0x7a23,0x76c3,0x82b7,0x8d9f,0x96c0,0x9739,
0x55e8,0x5428,0x62c2,0x6fc3,0x64cb,0x53a8,0x7ef3,0x88f9,
0x91e3,0x56b7,0x905c,0x6da8,0x76b1,0x8d4c,0x5993,0x7aed,
0x8116,0x77ff,0x5c39,0x4f10,0x90ca,0x7545,0x819d,0x54c4,
0x5938,0x5b55,0x55b7,0x5606,0x9556,0x8e5f,0x4ec6,0x5f0a,
0x6491,0x60f1,0x76ef,0x63a0,0x7089,0x88d9,0x59e8,0x60df,
0x6ec4,0x80a2,0x962e,0x8523,0x4f2a,0x6f54,0x4ffa,0x8c05,
0x596a,0x80a0,0x9493,0x840a,0x8caa,0x5265,0x6284,0x8de1,
0x8d5a,0x937e,0x7a4c,0x5320,0x96c1,0x62da,0x6fa1,0x5e16,
0x56ca,0x70db,0x7642,0x790e,0x50d1,0x6db5,0x9727,0x8fc8,

0x5fb9,0x9f7f,0x8b00,0x5d16,0x8c28,0x8258,0x4e19,0x72a7,
0x7e5e,0x7529,0x5f25,0x58fd,0x723a,0x9a37,0x5378,0x64d2,
0x502a,0x5e06,0x808c,0x7e6a,0x98fd,0x9cf4,0x503a,0x6627,
0x86d9,0x8f9f,0x5239,0x8ca2,0x62d0,0x80a4,0x96c7,0x5495,
0x58e2,0x8e72,0x4fef,0x543c,0x8e48,0x8bf1,0x64bf,0x9b44,
0x8015,0x9716,0x798d,0x7554,0x5925,0x60a6,0x8273,0x6490,
0x6372,0x5d50,0x5632,0x8d37,0x5401,0x752b,0x9742,0x64e0,
0x89bd,0x5bd3,0x8b6c,0x746a,0x9888,0x9b41,0x8df5,0x55a7,
0x658c,0x8d3e,0x632a,0x8f7f,0x6df9,0x51a5,0x5a07,0x5c65,
0x971c,0x6d78,0x6f47,0x6dd8,0x9326,0x8d50,0x6953,0x8cd3,
0x575f,0x515c,0x9882,0x5021,0x7344,0x5074,0x5937,0x7816,
0x6e14,0x5b9b,0x5c51,0x60b6,0x5bb5,0x5dfe,0x6b79,0x900d,
0x5ac2,0x6380,0x9709,0x54d1,0x55ac,0x9f52,0x997c,0x632b,
0x4fae,0x82ad,0x5dba,0x97fb,0x6ee9,0x727a,0x9489,0x88f8,
0x8e64,0x8d60,0x94c3,0x9081,0x7a1a,0x50a2,0x987d,0x6795,
0x8b7d,0x5a1f,0x8932,0x5d29,0x902e,0x50f5,0x916c,0x79e4,
0x8f68,0x54df,0x99db,0x7855,0x6734,0x78d5,0x5e05,0x61d2,
0x819c,0x517d,0x51c4,0x8a79,0x5583,0x730e,0x6500,0x574e,
0x9965,0x6bbc,0x5ab3,0x55d3,0x5eff,0x5147,0x934b,0x903b,
0x61fc,0x92b3,0x508d,0x8d29,0x9a84,0x84b8,0x7ed8,0x96ef,
0x7109,0x7948,0x64b0,0x4e2b,0x8667,0x604d,0x6670,0x95ca,
0x5857,0x9119,0x593e,0x9130,0x6085,0x4fde,0x8f5f,0x86ee,
0x640f,0x99d0,0x6398,0x8ced,0x4fd8,0x8350,0x62f7,0x8eac,
0x6b3d,0x6d29,0x51f3,0x9905,0x5f4c,0x8cab,0x9017,0x72ac,
0x7db1,0x7ff9,0x65a9,0x50da,0x58ae,0x9811,0x5415,0x8c79,
0x6ea2,0x8d08,0x8461,0x5f77,0x722a,0x6055,0x5c2c,0x6f8e,
0x62cb,0x5c4e,0x68ad,0x8c2d,0x683d,0x6467,0x584c,0x788c,
0x68fa,0x57ae,0x5824,0x51bb,0x79aa,0x9e9f,0x707d,0x9a86,
0x5e18,0x5075,0x6cfc,0x62f1,0x88d4,0x8b9a,0x6726,0x6292,
0x8404,0x8c26,0x5c09,0x5395,0x8da8,0x7737,0x4e5e,0x8e81,
0x9077,0x814a,0x9edb,0x7210,0x617e,0x9c8d,0x8650,0x4ed3,
0x79c3,0x5a77,0x7ed1,0x672d,0x764c,0x8235,0x803f,0x755c,
0x60bc,0x9e2d,0x7184,0x6feb,0x6f32,0x8bca,0x8ce4,0x5466,
0x54e8,0x7eb9,0x9d5d,0x5e9e,0x8ecc,0x9a9a,0x5858,0x55e4,
0x8c9e,0x895f,0x4f84,0x7955,0x8766,0x57d4,0x8b20,0x81a0,
0x905e,0x4f6c,0x54d7,0x69fd,0x4ea9,0x9ad2,0x6e20,0x561f,
0x8c0e,0x5006,0x7bee,0x88d8,0x6401,0x5cfb,0x53a2,0x868a,
0x5ae3,0x5faa,0x7792,0x6c90,0x5c4d,0x947d,0x56da,0x80bf,
0x810a,0x6c13,0x7830,0x5564,0x8fe6,0x5ec1,0x9db4,0x55aa,
0x4fa8,0x53e0,0x7051,0x75ab,0x5578,0x64c5,0x80c0,0x8d4b,
0x52df,0x8108,0x6073,0x7e8f,0x5a74,0x8e44,0x8165,0x714e,
0x664c,0x6afb,0x6e3e,0x6ed4,0x6bd9,0x7329,0x6963,0x5641,
0x8102,0x8c1c,0x6c27,0x8774,0x857e,0x545c,0x5bc7,0x6233,
0x9881,0x7a9c,0x884d,0x5132,0x687f,0x7942,0x7e6b,0x988a,
0x5140,0x8332,0x5631,0x9127,0x68df,0x9a73,0x9a30,0x59ec,
0x7cbd,0x53e1,0x5662,0x4f75,0x5243,0x80ba,0x9eef,0x566a,
0x6649,0x6487,0x9d28,0x4f83,0x6e3a,0x6caa,0x66a2,0x8d31,
0x90dd,0x7130,0x5291,0x56bc,0x602f,0x98c6,0x651c,0x7b77,
0x5992,0x87ba,0x83cc,0x58c7,0x559a,0x94a9,0x5102,0x7ad6,
0x60e9,0x803d,0x6eb6,0x7ee3,0x90e1,0x8bb6,0x7eb2,0x6ac3,
0x87fb,0x8ca9,0x8231,0x62d9,0x5a31,0x9e26,0x72ee,0x560e,

0x9699,0x7a9f,0x74f7,0x51db,0x9f9f,0x7fa1,0x711a,0x903e,
0x7a91,0x8972,0x8587,0x5ba0,0x7ea4,0x94fe,0x7aff,0x8b39,
0x853d,0x655e,0x72ed,0x6558,0x4f3a,0x94ed,0x8cca,0x86db,
0x8fc4,0x68b3,0x6e0a,0x83bd,0x71e6,0x7faf,0x4fb6,0x6524,
0x886c,0x7a57,0x7fa8,0x8b0e,0x7dbf,0x7f05,0x7c9e,0x8ce0,
0x6402,0x918b,0x7c64,0x549a,0x533f,0x9a87,0x8105,0x50fb,
0x761f,0x6bcb,0x81a8,0x7f1a,0x547b,0x707f,0x7d10,0x8206,
0x55b2,0x8a60,0x818f,0x5375,0x83c1,0x65a7,0x58f9,0x6514,
0x97f5,0x7a83,0x819a,0x9a5f,0x55e1,0x66a8,0x80f3,0x748b,
0x6123,0x7693,0x9877,0x75de,0x673d,0x6cb8,0x5308,0x5edf,
0x63ea,0x6687,0x4fa6,0x6577,0x8178,0x75d2,0x7fe9,0x6346,
0x8038,0x9171,0x65f1,0x5009,0x58f6,0x9661,0x5bee,0x8b0a,
0x6652,0x8cbf,0x9a74,0x58f3,0x6ca7,0x79a6,0x8d1e,0x818a,
0x8e34,0x6bef,0x860b,0x71e5,0x5b7d,0x64bc,0x4fcf,0x8f85,
0x79be,0x7538,0x595a,0x8511,0x5d17,0x8d26,0x76ea,0x5be5,
0x66c6,0x8403,0x532f,0x8a98,0x5366,0x557c,0x6361,0x60ed,
0x599e,0x5636,0x553e,0x8b19,0x7caa,0x9470,0x9215,0x6ee5,
0x5315,0x582f,0x76e7,0x6e83,0x9a7c,0x96b6,0x61a4,0x879e,
0x4e52,0x8bc0,0x65f7,0x5962,0x7d0b,0x744b,0x56c2,0x5256,
0x5c34,0x6bd3,0x70ad,0x7f34,0x5f7f,0x5450,0x7375,0x88b1,
0x52f3,0x82f9,0x61c7,0x74ca,0x51cd,0x8782,0x78ca,0x7a4e,
0x8ef8,0x540b,0x5514,0x5986,0x6dc0,0x8721,0x58e4,0x851a,
0x6a11,0x5c61,0x6273,0x6f51,0x51f8,0x970e,0x9f90,0x63e3,
0x5242,0x8c41,0x7c98,0x608d,0x9285,0x9aa4,0x95a9,0x6177,
0x8be7,0x7cfe,0x53e2,0x819b,0x5e62,0x8f96,0x55fd,0x934a,
0x6e4a,0x960e,0x7ca5,0x85b0,0x87d1,0x63b0,0x62e2,0x7a3c,
0x8463,0x7784,0x7728,0x80e7,0x6d95,0x7977,0x8bbd,0x9ecf,
0x6583,0x94ee,0x9a55,0x5983,0x79c9,0x511f,0x60d5,0x62e3,
0x9187,0x78b3,0x84e6,0x6869,0x540f,0x8569,0x6f64,0x8c23,
0x695e,0x5cb1,0x9913,0x5760,0x6ede,0x7011,0x7095,0x4f47,
0x7504,0x8bf5,0x659f,0x85af,0x6fd5,0x4f36,0x852c,0x75a4,
0x507d,0x8e10,0x7eb1,0x8ca7,0x8b2c,0x7a3d,0x83b9,0x8854,
0x8be1,0x817b,0x8d2c,0x99c1,0x6df5,0x717d,0x99b3,0x635e,
0x6405,0x8098,0x4f1e,0x94f8,0x8eaf,0x7b19,0x73c2,0x6eaf,
0x70b3,0x65ac,0x63c9,0x6b7c,0x8a6d,0x82bd,0x6631,0x8042,
0x6dcc,0x5fff,0x9980,0x70eb,0x821c,0x4ed1,0x949e,0x77a5,
0x6d46,0x72f8,0x5c94,0x4ed5,0x625b,0x8543,0x6893,0x5a7f,
0x7c60,0x7bf7,0x5960,0x8a1d,0x6666,0x985b,0x6cae,0x745b,
0x8335,0x7bab,0x64b2,0x776c,0x9e45,0x917f,0x53e8,0x572d,
0x7662,0x8328,0x8a57,0x6e85,0x5179,0x6363,0x5029,0x5431,
0x680b,0x6da9,0x7980,0x7f06,0x82df,0x70c1,0x8bc8,0x61ff,
0x6e1d,0x6026,0x8db4,0x8d66,0x53ae,0x631f,0x51b6,0x6115,
0x8cc4,0x9ad3,0x7dfb,0x9068,0x72e1,0x542d,0x88f3,0x9952,
0x7426,0x82db,0x6d85,0x6413,0x7422,0x95d6,0x8af7,0x9791,
0x8a3a,0x8ae7,0x6399,0x921e,0x8993,0x701f,0x8654,0x7eba,
0x68a2,0x92ea,0x61f8,0x70ab,0x9524,0x4e53,0x5a34,0x5151,
0x8a6e,0x51f9,0x8fab,0x6c41,0x650f,0x58fa,0x9cc4,0x76cf,
0x4e4d,0x7115,0x7076,0x5815,0x6dd2,0x7c3f,0x674f,0x89c5,
0x8085,0x8c10,0x6f13,0x5955,0x70d8,0x6ffe,0x7e96,0x91ac,
0x7d6e,0x618e,0x5885,0x5e9a,0x8f3b,0x79a7,0x94a5,0x634d,
0x5dcd,0x7eee,0x9713,0x79bd,0x62ef,0x66dd,0x5bc5,0x5ac9,

```
 0x5bf5,0x60b8,0x6f01,0x5384,0x588a,0x7ef8,0x5201,0x7e2b,
 0x99ff,0x763e,0x7736,0x53e9,0x901e,0x8e66,0x6020,0x57ab,
 0x6d47,0x5f64,0x60d8,0x52d8,0x5f8a,0x8046,0x618b,0x95f5,
 0x99dd,0x8549,0x50d5,0x758a,0x62ed,0x55dc,0x7b8f,0x82b8,
 0x7194,0x524e,0x840e,0x589c,0x6912,0x8513,0x592d,0x766e,
 0x7efd,0x881f,0x5f98,0x725f,0x96a7,0x68b5,0x79b1,0x772f,
 0x5162,0x6a61,0x914c,0x749e,0x7c72,0x5f13,0x5a1b,0x98b1,
 0x9798,0x7409,0x773a,0x64ab,0x9e20,0x5098,0x5b0c,0x932b,
 0x77bb,0x6c85,0x6ca6,0x7f15,0x889c,0x6c8c,0x797a,0x79bf,
 0x6dea,0x7682,0x525d,0x759a,0x841d,0x9ae6,0x795f,0x6f9c,
 0x8700,0x53a5,0x6e67,0x6e17,0x500f,0x7a98,0x61c8,0x6043,
 0x63fd,0x82b9,0x8f69,0x6cd3,0x836b,0x7d43,0x78da,0x6c83
};

#define UNIHAN_REORDER_BASE 0X5000

u_code_point reorder_unihan(u_code_point s) {
        u_code_point i=UNIHAN_REORDER_BASE;
        int k=0;
        for(k=0; k<UH; k++,i++) {
           if(s == unihan_freq[k]) { return i; };
        };
        k=(s - UNIHAN_REORDER_BASE);
        if( k>=0 && k<UH) {
           return unihan_freq[k];
        };
        return s;
}

u_code_point restore_order_unihan(u_code_point z) {
        u_code_point i=UNIHAN_REORDER_BASE;
        int k;
        k=(z - UNIHAN_REORDER_BASE);
        if( k>=0 && k<UH) {
           return unihan_freq[k];
        };
        for(k=0; k<UH; k++,i++) {
           if(z == unihan_freq[k]) { return i; };
        };
        return z;
}


#define KATAKANA_REORDER_BASE 0X30A0
//KATAKANA reorder by frequency in Japanese Business
#define KK 96
u_code_point katakana_freq[KK] = {
0x30f3,0x30eb,0x30b9,0x30c8,0x30a2,0x30a4,0x30e9,0x30ea,
0x30af,0x30c3,0x30fc,0x30b7,0x30b8,0x30e7,0x30ec,0x30b0,
0x30d5,0x30d7,0x30df,0x30c4,0x30ef,0x30a8,0x30cb,0x30e1,
0x30ab,0x30c6,0x30b3,0x30dd,0x30d9,0x30cf,0x30c9,0x30a6,
```

```
0x30bb,0x30ce,0x30ca,0x30e0,0x30ed,0x30bf,0x30c1,0x30d0,
0x30b4,0x30dc,0x30bd,0x30cd,0x30e2,0x30d3,0x30b5,0x30ad,
0x30b1,0x30b2,0x30bc,0x30e8,0x30e5,0x30aa,0x30cc,0x30a3,
0x30d6,0x30de,0x30a1,0x30a5,0x30a7,0x30a9,0x30ac,0x30ae,
0x30b6,0x30ba,0x30be,0x30c0,0x30c2,0x30c5,0x30c7,0x30d1,
0x30d2,0x30d4,0x30d8,0x30da,0x30db,0x30e3,0x30e4,0x30e6,
0x30ee,0x30f0,0x30f1,0x30f2,0x30f4,0x30f5,0x30f6,0x30f7,
0x30f8,0x30f9,0x30fa,0x30fb,0x30fd,0x30fe,0x30ff,0x30a0,
};

u_code_point reorder_katakana(u_code_point s) {
        u_code_point i=KATAKANA_REORDER_BASE;
        int k=0;
        for(k=0; k<KK; k++,i++) {
           if(s == katakana_freq[k]) { return i; };
        };
        return s; // not reached here
}

u_code_point restore_order_katakana(u_code_point z) {
        return katakana_freq[z - KATAKANA_REORDER_BASE];
}


#define HINDI_REORDER_BASE 0X0900
//HINDI reorder by frequency in Japanese Business
#define HD 113
u_code_point hindi_freq[HD] = {
0x0902,0x093f,0x0940,0x0947,0x0948,0x094b,0x094d,0x0915,
0x0917,0x0932,0x0938,0x0939,0x0924,0x0926,0x0928,0x092c,
0x092f,0x0900,0x0901,0x0903,0x0904,0x0916,0x0918,0x0919,
0x091a,0x091b,0x091c,0x091d,0x091e,0x091f,0x0920,0x0921,
0x0922,0x0923,0x0925,0x0927,0x0929,0x092a,0x092b,0x092d,
0x092e,0x0930,0x0931,0x0933,0x0934,0x0935,0x0936,0x0937,
0x093a,0x093b,0x093c,0x093d,0x093e,0x0941,0x0942,0x0943,
0x0944,0x0945,0x0946,0x0949,0x094a,0x094c,0x094e,0x094f,
0x0950,0x0951,0x0952,0x0953,0x0954,0x0955,0x0956,0x0957,
0x0958,0x0959,0x095a,0x095b,0x095c,0x095d,0x095e,0x095f,
0x0960,0x0961,0x0962,0x0963,0x0964,0x0965,0x0966,0x0967,
0x0968,0x0969,0x096a,0x096b,0x096c,0x096d,0x096e,0x096f,
0x0905,0x0906,0x0907,0x0908,0x0909,0x090a,0x090b,0x090c,
0x090d,0x090e,0x090f,0x0910,0x0911,0x0912,0x0913,0x0914,
0x0970
};

u_code_point reorder_hindi(u_code_point s) {
        u_code_point i=HINDI_REORDER_BASE;
        int k=0;
        for(k=0; k<HD; k++,i++) {
           if(s == hindi_freq[k]) { return i; };
        };
```

```
        return s; // not reached here
}

u_code_point restore_order_hindi(u_code_point z) {
        return hindi_freq[z - HINDI_REORDER_BASE];
}


#define MAPCHAR(x,A,B,bytes) if(A<=x && x< (A+bytes)) \
        return(x+(B-A)); if(B<=x && x< (B+bytes)) return(x+(A-B))
#define MAP16BL(x,A,B,block) if(A<=x && x< (A+(block<<4))) \
        return(x+(B-A)); if(B<=x && x< (B+(block<<4))) \
        return(x+(A-B))

u_code_point reorder_latins(u_code_point s) {
        MAP16BL(s,0x0100,0x0000,3); // Latin Extension A
        MAP16BL(s,0x0130,0x0080,2);
        MAP16BL(s,0x0150,0x00A0,1);
        MAP16BL(s,0x0300,0x00B0,3); // Combining Diacritical Marks
        MAPCHAR(s,0x0070,0x0060,1); // p,`
        MAPCHAR(s,0x0072,0x006A,1); // r,j
        MAPCHAR(s,0x0073,0x006B,1); // s,k
        MAPCHAR(s,0x0074,0x0066,1); // t,f
        MAPCHAR(s,0x0075,0x0067,1); // u,g
        MAPCHAR(s,0x0050,0x0040,1); // P,@  UPPER
        MAPCHAR(s,0x0052,0x004A,1); // R,J  UPPER
        MAPCHAR(s,0x0053,0x004B,1); // S,K  UPPER
        MAPCHAR(s,0x0054,0x0046,1); // T,F  UPPER
        MAPCHAR(s,0x0055,0x0047,1); // U,G  UPPER
        MAPCHAR(s,0x0160,0x003A,6); // Latin Extension A
        MAPCHAR(s,0x0166,0x005B,5);
        MAPCHAR(s,0x016B,0x007B,5);
        return s;
}

u_code_point restore_order_latins(u_code_point z) {
        return reorder_latins(z);
}

u_code_point reorder(u_code_point s) {
        if(isHANGUL(s)) return reorder_hangul(s);
        if(isUNIHAN(s)) return reorder_unihan(s);
        if(isKATAKANA(s)) return reorder_katakana(s);
        if(isHINDI(s)) return reorder_hindi(s);
        if(isLatins(s)) return reorder_latins(s);
        return s;
}
u_code_point restore_order(u_code_point s) {
        if(isHANGUL(s)) return restore_order_hangul(s);
        if(isUNIHAN(s)) return restore_order_unihan(s);
        if(isKATAKANA(s)) return restore_order_katakana(s);
```

```
        if(isHINDI(s)) return restore_order_hindi(s);
        if(isLatins(s)) return restore_order_latins(s);
        return s;
}
```

```
/* update(refpoint,style,n,k) updates refpoint[1..3] and *style */
/* based on n (the most recent code point) and k (the number of */
/* base-32 characters used to encode it).                       */

static void update( u_code_point refpoint[6], unsigned int *style,
                    u_code_point n, unsigned int k                 )
{
  *style = k < 3 ? 0 : k > 3 ? 1 : *style;
  refpoint[1] = (n >> 4) << 4;
  if (k > 2) refpoint[2] = n - 0xA0 < 0xE0 ? 0xA0 : (n >> 8) << 8;
  if (k > 3) refpoint[3] = n - 0x3000 < 0x7000 ? 0x4E00 :
    *style == 1 && n - 0xA000 < 0x3800 ? 0x8800 : (n >> 12) << 12;
}

/* Main encode function: */

enum amc_ace_status amc_ace_w_encode(
  unsigned int input_length,
  const u_code_point input[],
  const unsigned char uppercase_flags[],
  unsigned int *output_size,
  char output[] )
{
  unsigned int style, literal, max_out, in, out, k, j;
  u_code_point n, delta;
  const u_code_point maxdelta[2][6] =
  {{0,0xF,0xFF,0xFFF,0xFFFF,0xFFFFF}, {0,0,0xFF,0x4FFF,0xFFFF,0xFFFFF}};
  char shift;

  /* Initialize the state: */

  u_code_point refpoint[6] = {0, 0xE0, 0xA0, 0, 0, 0x10000};

  style = literal = 0;
  max_out = *output_size;

  for (in = out = 0;  in < input_length;  ++in) {

    /* At the start of each iteration, in and out are the number of */
    /* items already input/output, or equivalently, the indices of  */
    /* the next items to be input/output.                           */
```

```
n = input[in];
/* Check the code point range to avoid array bounds errors later: */
if (n > 0x10FFFF) return amc_ace_bad_input;

if (n == 0x2D) {
  /* Hyphen-minus is doubled. */
  if (max_out - out < 2) return amc_ace_big_output;
  output[out++] = 0x2D;
  output[out++] = 0x2D;
}
else if ( n <= 122 && ( n >= 97 || n == 45 ||
          (n >= 48 && n <= 57) || (n >= 65 && n <= 90) ) ) {
  /* Encode an LDH character literally. */
  if (max_out - out < 1 + !literal) return amc_ace_big_output;
  /* Switch to literal mode if necessary: */
  if (!literal) output[out++] = 0x2D;
  literal = 1;
  output[out++] = n;
}
else {
  /* Encode a non-LDH character using base-32.           */
  /* First compute the number of base-32 characters (k): */

  n = reorder(n); // ADDED *****************

  for (k = 1 + style;  ;  ++k) {
    delta = n - refpoint[k];
    if (delta <= maxdelta[style][k]) break;
  }

  if (max_out - out < k + literal) return amc_ace_big_output;
  /* Switch to base-32 mode if necessary: */
  if (literal) output[out++] = 0x2D;
  literal = 0;
  shift = uppercase_flags && uppercase_flags[in] ? 32 : 0;

  /* Check for the extended delta of style 1 window 3: */

  if (k == 3 && delta >= 0x1000) {
    /* The top 16k of window 3 is encoded as 0xxxx xxxxx xxxxx. */
    delta -= 0x1000;
    output[out++] = base32[delta >> 10] - shift;
    output[out++] = base32[(delta >> 5) & 0x1F];
    output[out++] = base32[delta & 0x1F];
  }
  else {
    /* Each quintet has the form 1xxxx except the last is 0xxxx. */
    /* Computing the base-32 digits in reverse order is easiest. */

    out += k;
    output[out - 1] = base32[delta & 0xF] - shift;
```

```
        for (j = 2;  j <= k;  ++j) {
          delta >>= 4;
          output[out - j] = base32[0x10 | (delta & 0xF)];
        }
      }

      update(refpoint, &style, n, k);
    }
  }

  /* Append the null terminator: */
  if (max_out - out < 1) return amc_ace_big_output;
  output[out++] = 0;

  *output_size = out;
  return amc_ace_success;
}

/* Main decode function: */

enum amc_ace_status amc_ace_w_decode(
  enum case_sensitivity case_sensitivity,
  char scratch_space[],
  const char input[],
  unsigned int *output_length,
  u_code_point output[],
  unsigned char uppercase_flags[] )
{
  u_code_point q, delta;
  char c;
  unsigned int style, literal, max_out, in, out, k, scratch_size;
  enum amc_ace_status status;

  /* Initialize the state: */

  u_code_point refpoint[6] = {0, 0xE0, 0xA0, 0, 0, 0x10000};

  style = literal = 0;
  max_out = *output_length;

  for (c = input[in = 0], out = 0;  c != 0;  c = input[++in], ++out) {

    /* At the start of each iteration, in and out are the number of   */
    /* items already input/output, or equivalently, the indices of    */
    /* the next items to be input/output. c is the same as input[in]  */
    /* except when "extra" characters have been consumed (see below). */

    if (c == 0x2D && input[in + 1] != 0x2D) {
      /* Unpaired hyphen-minus toggles mode. */
      literal = !literal;
      c = input[++in];
```

```
  }

  if (max_out - out < 1) return amc_ace_big_output;

  if (c == 0x2D) {
    /* Double hyphen-minus represents a hyphen-minus. */
    ++in;
    output[out] = 0x2D;
  }
  else {
    if (literal) output[out] = c;
    else {
      /* Decode a base-32 sequence.                  */
      /* First decode quintets until 0xxxx is found: */

      for (delta = 0, k = 1;  ;  c = input[++in], ++k) {
        q = base32_decode(c);
        if (q == base32_invalid || k > 5) return amc_ace_bad_input;
        delta = (delta << 4) | (q & 0xF);
        if (q >> 4 == 0) break;
      }

      if (style == 1 && k == 1) {
        /* Style 1 has no window 1, so it must be the extended */
        /* delta of window 3, encoded as 0xxxx xxxxx xxxxx.    */
        /* Consume the two "extra" characters:                 */

        for (;  k < 3;  ++k) {
          q = base32_decode(input[++in]);
          if (q == base32_invalid) return amc_ace_bad_input;
          delta = (delta << 5) | q;
        }

        delta += 0x1000;
      }

      output[out] = refpoint[k] + delta;
      update(refpoint, &style, output[out], k);

      output[out] = restore_order(output[out]); // ADDED

    }
  }

  /* Case of last non-extra character determines uppercase flag: */
  if (uppercase_flags) uppercase_flags[out] = c >= 65 && c <= 90;
}

/* Enforce the uniqueness of the encoding by re-encoding */
/* the output and comparing the result to the input:     */

scratch_size = ++in;
```

```
  status = amc_ace_w_encode(out, output, uppercase_flags,
                            &scratch_size, scratch_space);
  if (status != amc_ace_success || scratch_size != in ||
      unequal(case_sensitivity, scratch_space, input)
     ) return amc_ace_bad_input;

  *output_length = out;
  return amc_ace_success;
}


/*****************************************************************/
/* Wrapper for testing (would normally go in a separate .c file): */

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* For testing, we'll just set some compile-time limits rather than */
/* use malloc(), and set a compile-time option rather than using a  */
/* command-line option.                                             */

enum {
  unicode_max_length = 256,
  ace_max_size = 256,
  test_case_sensitivity = case_insensitive
                          /* suitable for host names */
};


static void usage(char **argv)
{
  fprintf(stderr,
    "%s -e reads code points and writes an AMC-ACE-W string.\n"
    "%s -d reads an AMC-ACE-W string and writes code points.\n"
    "Input and output are plain text in the native character set.\n"
    "Code points are in the form u+hex separated by whitespace.\n"
    "An AMC-ACE-W string is a newline-terminated sequence of LDH\n"
    "characters (without any signature).\n"
    "The case of the u in u+hex is the force-to-uppercase flag.\n"
    , argv[0], argv[0]);
  exit(EXIT_FAILURE);
}

static void fail(const char *msg)
{
  fputs(msg,stderr);
  exit(EXIT_FAILURE);
}

static const char too_big[] =
```

```
  "input or output is too large, recompile with larger limits\n";
static const char invalid_input[] = "invalid input\n";
static const char io_error[] = "I/O error\n";


/* The following string is used to convert LDH      */
/* characters between ASCII and the native charset: */

static const char ldh_ascii[] =
  "................"
  "................"
  "............-..."
  "0123456789......"
  ".ABCDEFGHIJKLMNO"
  "PQRSTUVWXYZ....."
  ".abcdefghijklmno"
  "pqrstuvwxyz";


int main(int argc, char **argv)
{
  enum amc_ace_status status;
  int r;
  char *p;

  if (argc != 2) usage(argv);
  if (argv[1][0] != '-') usage(argv);
  if (argv[1][2] != 0) usage(argv);

  if (argv[1][1] == 'e') {
    u_code_point input[unicode_max_length];
    unsigned long codept;
    unsigned char uppercase_flags[unicode_max_length];
    char output[ace_max_size], uplus[3];
    unsigned int input_length, output_size, i;

    /* Read the input code points: */

    input_length = 0;

    for (;;) {
      r = scanf("%2s%lx", uplus, &codept);
      if (ferror(stdin)) fail(io_error);
      if (r == EOF || r == 0) break;

      if (r != 2 || uplus[1] != '+' || codept > (u_code_point)-1) {
        fail(invalid_input);
      }

      if (input_length == unicode_max_length) fail(too_big);

      if (uplus[0] == 'u') uppercase_flags[input_length] = 0;
```

```
    else if (uplus[0] == 'U') uppercase_flags[input_length] = 1;
    else fail(invalid_input);

    input[input_length++] = codept;
  }

  /* Encode: */

  output_size = ace_max_size;
  status = amc_ace_w_encode(input_length, input, uppercase_flags,
                            &output_size, output);
  if (status == amc_ace_bad_input) fail(invalid_input);
  if (status == amc_ace_big_output) fail(too_big);
  assert(status == amc_ace_success);

  /* Convert to native charset and output: */

  for (p = output;  *p != 0;  ++p) {
    i = *p;
    assert(i <= 122 && ldh_ascii[i] != '.');
    *p = ldh_ascii[i];
  }

  r = puts(output);
  if (r == EOF) fail(io_error);
  return EXIT_SUCCESS;
}

if (argv[1][1] == 'd') {
  char input[ace_max_size], scratch[ace_max_size], *pp;
  u_code_point output[unicode_max_length];
  unsigned char uppercase_flags[unicode_max_length];
  unsigned int input_length, output_length, i;

  /* Read the AMC-ACE-W input string and convert to ASCII: */

  fgets(input, ace_max_size, stdin);
  if (ferror(stdin)) fail(io_error);
  if (feof(stdin)) fail(invalid_input);
  input_length = strlen(input);
  if (input[input_length - 1] != '\n') fail(too_big);
  input[--input_length] = 0;

  for (p = input;  *p != 0;  ++p) {
    pp = strchr(ldh_ascii, *p);
    if (pp == 0) fail(invalid_input);
    *p = pp - ldh_ascii;
  }

  /* Decode: */

  output_length = unicode_max_length;
```

```
    status = amc_ace_w_decode(test_case_sensitivity, scratch, input,
                              &output_length, output, uppercase_flags);
    if (status == amc_ace_bad_input) fail(invalid_input);
    if (status == amc_ace_big_output) fail(too_big);
    assert(status == amc_ace_success);

    /* Output the result: */

    for (i = 0;  i < output_length;  ++i) {
      r = printf("%s+%04lX\n",
                 uppercase_flags[i] ? "U" : "u",
                 (unsigned long) output[i] );
      if (r < 0) fail(io_error);
    }

    return EXIT_SUCCESS;
  }

  usage(argv);
  return EXIT_SUCCESS;  /* not reached, but quiets compiler warning */
}

/* end of lamcw.c */
```