CISP 440

Austin Smothers

Assignment 1

/\*

This program builds on the bases.cpp code supplied by

Professor Ross.

\*/

/\*

Converts an integer from one base to another.

Dan Ross

Jan 2013

Supports any base between 3 and 36. Integers are represented

as strings consisting of a subset of the following characters:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

EXAMPLE: 42 base10 = 101010 base2 = 2A base16 = 60 base 7 = 16 base36

Also converts 3-letter base 4 sequences of the following digits

GATC

3210

EXAMPLES:

CCC = 0x4^2 + 0x4^1 + 0x4^0 = 0 + 0 + 0 = 0

GGG = 3x4^2 + 3x4^1 + 3x4^0 = 48 + 12 + 3 = 63

ACT = 2x4^2 + 0x4^1 + 1x4^0 = 32 + 0 + 1 = 33

\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#pragma warning( disable : 4996)

#pragma warning( disable : 4244)

/\*

Calculates base raised to the power exp

\*/

int my\_pow(int base, int exp)

{

int x = 1;

for (int i = 0; i < exp; i++)

x \*= base;

return x;

}

/\*

Converts an alphanumeric string 's' representation of a base 'base' number

to internal binary format.

Returns the value as a 32 bit integer

\*/

long my\_atoi(char \* s, int base)

{

long i = strtol(s, nullptr, base);

/\*

strtol takes the string (s) provided by the user

then converts it into base 10 from format

base (base) provided from passing the argument

follows format: const char \* str, char \*\* endptr (nullptr), int base)

\*/

return i;

}

/\*

Converts a 32 bit integer 'n' into an alphanumeric string 'sOut'

representation of a base 'base' number

\*/

void my\_itoa(long n, char \* sOut, int base)

{

itoa(n, sOut, base);

/\*

converts an integer value to an unsigned string

exception: will display a signed string in base 10

follows format: itoa(int value, char \* str, int base)

\*/

}

/\*

Converts an integer to base 4 codon format

\*/

void itoDNAcodon(char \* codon, int i)

{

itoa(i, codon, 4);

//since DNA is coded in base 4, you simply need

//a start number and you can use itoa to convert

//the codon back to base 4

for (int n = (sizeof(codon)); n >= 0; n--)

{

switch (codon[n])

{

case '3':

codon[n] = 'G'; break;

case '2':

codon[n] = 'A'; break;

case '1':

codon[n] = 'T'; break;

case '0':

codon[n] = 'C'; break;

}

}

/\*

Having the base 4 number, all that was left was to

convert that number back into GATC format, which can be

done with a switch statement and a for loop, iterating once

per array member. In this case, we use sizeof(codon) to get

the size of the codon array used in the for loop

It is also important to read the entire array including element 0,

or you won't convert the entire base 4 long value into a string

\*/

}

/\*

Returns the base 4 value of a codon

\*/

int DNAcodontoi(char codon[4])

{

int d0, d1, d2, value;

switch (codon[2])

{

case 'C':

d0 = 0; break;

case 'T':

d0 = 1; break;

case 'A':

d0 = 2; break;

case 'G':

d0 = 3; break;

}

switch (codon[1])

{

case 'C':

d1 = 0; break;

case 'T':

d1 = 1; break;

case 'A':

d1 = 2; break;

case 'G':

d1 = 3; break;

}

switch (codon[0])

{

case 'C':

d2 = 0; break;

case 'T':

d2 = 1; break;

case 'A':

d2 = 2; break;

case 'G':

d2 = 3; break;

}

// the base 4 value of the codon

value = d2 \* 16 + d1 \* 4 + d0;

return value;

}

/\*

Adds 2 string representations of 2 base N numbers

s1 and s2 are operands, after the function returns, sum will contain the result

\*/

void addInBase(char s1[], char s2[], char sum[], int base)

{

long i = strtol(s1, nullptr, base);

long j = strtol(s2, nullptr, base);

long k = i + j;

/\*

To add the two strings together, they are first converted

from whatever base they are in to base 10, then added and

stored in the long variable k

\*/

my\_itoa(k, sum, base);

//k is then converted back to a string using my\_itoa

}

void main()

{

char sIn[] = "16";

char sOut[80];

long n = 0;

int baseNum = 0;

//baseNum added as a variable for ease of testing

/\*Round 1 of baseNum to base 10 to baseNum conversions

\*/

// convert some wacky base to internal representation

baseNum = 36;

n = my\_atoi(sIn, baseNum);

printf("%s base %d is %d base 10\n", sIn, baseNum, n);

// convert back to some wacky base

baseNum = 11;

my\_itoa(n, sOut, baseNum);

printf("%d base 10 is %s base %d\n", n, sOut, baseNum);

/\*Round 2 of baseNum to base 10 to baseNum conversions

\*/

// convert some wacky base to internal representation

baseNum = 13;

n = my\_atoi(sIn, baseNum);

printf("%s base %d is %d base 10\n", sIn, baseNum, n);

// convert back to some wacky base

baseNum = 24;

my\_itoa(n, sOut, baseNum);

printf("%d base 10 is %s base %d\n", n, sOut, baseNum);

/\*Round 3 of baseNum to base 10 to baseNum conversions

\*/

// convert some wacky base to internal representation

baseNum = 8;

n = my\_atoi(sIn, baseNum);

printf("%s base %d is %d base 10\n", sIn, baseNum, n);

// convert back to some wacky base

baseNum = 17;

my\_itoa(n, sOut, baseNum);

printf("%d base 10 is %s base %d\n", n, sOut, baseNum);

/\* Round 1 with DNA

\*/

char muhGenes[4] = "ACT";

n = DNAcodontoi(muhGenes);

printf("%s in DNA is %d base 10\n", muhGenes, n);

// back to DNA cuz we can

char buffy[4];

itoDNAcodon(buffy, n);

printf("%d base 10 is %s in DNA\n", n, buffy);

/\* Round 2 with DNA

\*/

char duhGenes[4] = "TAG";

//a new string variable must be declared here since muhGenes

//is of type const char once defined

n = DNAcodontoi(duhGenes);

printf("%s in DNA is %d base 10\n", duhGenes, n);

// back to DNA cuz we can

itoDNAcodon(buffy, n);

printf("%d base 10 is %s in DNA\n", n, buffy);

/\* Round 3 with DNA

\*/

char buhGenes[4] = "GAT";

//I wanted to do GATTACA, but Prof's code only decodes 3 letter sequences

//and my head cold means I'm feeling to lazy to fix it \*insert more excuses here\*

//a new string variable must be declared here since muhGenes

//is of type const char once defined

n = DNAcodontoi(buhGenes);

printf("%s in DNA is %d base 10\n", buhGenes, n);

// back to DNA cuz we can

itoDNAcodon(buffy, n);

printf("%d base 10 is %s in DNA\n", n, buffy);

/\*Round 1 adding strings

\*/

// test adding strings in arbitrary bases

char s1[] = "ABBA";

char s2[] = "ACDC";

char sum[32];

baseNum = 16;

addInBase(s1, s2, sum, baseNum);

printf("%s + %s in base %d is %s\n", s1, s2, baseNum, sum);

/\*Round 2 adding strings

\*/

// test adding strings in arbitrary bases

char s3[] = "R2D2";

char s4[] = "C3P0";

baseNum = 32;

addInBase(s3, s4, sum, baseNum);

printf("%s + %s in base %d is %s\n", s3, s4, baseNum, sum);

/\*Round 3 adding strings

\*/

// test adding strings in arbitrary bases

char s5[] = "DARTH";

char s6[] = "VADER";

baseNum = 32;

addInBase(s5, s6, sum, baseNum);

printf("%s + %s in base %d is %s\n", s5, s6, baseNum, sum);

}

