CISP440

Joseph Morgan

Homework 1

Output

This output was written directly to an output.txt file via:

$: ./base_conversion_test.out > output.txt

```
Testing my_atoi():
Converting 14 in base 25 to base 10: 29
Converting FA in base 16 to base 10: 250
Converting 63 in base 29 to base 10: 177
Converting 2G in base 20 to base 10: 56
Converting Z9R in base 36 to base 10: 45711
Converting 72 in base 18 to base 10: 128


Testing my_itoa():
Converting 210 in base 10 to base 25: 8A
Converting 87 in base 10 to base 16: 57
Converting 714 in base 10 to base 29: OI
Converting 93 in base 10 to base 20: 4D
Converting 34 in base 10 to base 36: Y
Converting 218 in base 10 to base 18: C2


Testing itocodon():
Converting 18 to a DNA Codon: CTCA
Converting 23 to a DNA Codon: CTTG
Converting 10 to a DNA Codon: CCAA
Converting 29 to a DNA Codon: CTGT
Converting 34 to a DNA Codon: CACA
Converting 26 to a DNA Codon: CTAA


Testing add_in_base():
Adding 14 and FA in base 25: GE
Adding 63 and 2G in base 29: 8J
Adding Z9R and 72 in base 36: ZGT
```

Source Code

Test Main:

```
#include <stdio.h>
#include "./base_conversion.h"

void test_my_atoi();
void test_my_itoa();
void test_itocodon();
void test_add_in_base();

char* alpha_values[6] = {"14", "FA", "63", "2G", "Z9R", "72"};
const int int_values[6] = {210, 87, 714, 93, 34, 218};
const int bases[6] = {25, 16, 29, 20, 36, 18};

int main() {
	test_my_atoi();
	test_my_itoa();
	test_itocodon();
	test_add_in_base();
}

void test_my_atoi() {
	long result;

	printf("Testing my_atoi(): \n");
	for (int i = 0; i < 6; ++i) {
		result = my_atoi(alpha_values[i], bases[i]);
		printf("Converting %s in base %i to base 10: %li\n", alpha_values[i], bases[i],
result);
	}
}

void test_my_itoa() {
	char result[MAX_INPUT_LENGTH];

	printf("\n\nTesting my_itoa():\n");
	for (int i = 0; i < 6; ++i) {
		my_itoa(int_values[i], result, bases[i]);
		printf("Converting %i in base 10 to base %i: %s\n", int_values[i], bases[i], result);
	}
}

void test_itocodon() {
	char codon[4];
	printf("\n\nTesting itocodon(): \n");
	for (int i = 0; i < 6; ++i) {
		itocodon(codon, (int_values[i] % 64));
		printf("Converting %i to a DNA Codon: %s\n", (int_values[i] % 64), codon);
	}
}

void test_add_in_base() {
	char sum[MAX_INPUT_LENGTH];

	printf("\n\nTesting add_in_base(): \n");
	for (int i = 0; i <= 5; i += 2) {
		add_in_base(alpha_values[i], alpha_values[i+1], sum, bases[i]);
		printf("Adding %s and %s in base %i: %s\n", alpha_values[i], alpha_values[i+1],
bases[i], sum);
	}
}
```

Implementation:

```
#include <stdlib.h>
#include <stdio.h>
#include "./base_conversion.h"

const int MAX_BASE = 36;
const char char_lookup [MAX_BASE + 1] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A',
'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
'V', 'W', 'X', 'Y', 'Z', '-'};

int my_pow(int base, int exp) {
	int x = 1;

	for (int i = 0; i < exp; i++) {
		x *= base;
	}

	return x;
}

long my_atoi(char* s, int base) {
	long i_number = 0;
	unsigned len = -1;
	bool is_negative = false;
	if (s[0] == '-') {
		is_negative = true;
		s = &s[1];
	}

	while (s[++len] != '\0');

	for (int i = (len - 1), j = 0; i >= 0; --i, ++j) {
		if (s[i] >= 48 && s[i] <= 57) { // if s[i] is between characters '0' and '9'
			i_number += (s[i] - 48) * my_pow(base, j);
		} else if (s[i] >= 65 && s[i] <= 90) { // if s[i] is between 'A' and 'Z'
			i_number += (s[i] - 55) * my_pow(base, j);
		} else {
			printf("Parsing Error: Bad Character: %c\n", s[i]);
			exit(1);
		}
	}

	return (is_negative ? i_number * -1 : i_number);
}

void my_itoa(long n, char* sOut, int base) {
	int values_in_reverse[100];
	int stepper = 0;
	bool is_negative = false;

	if (n < 0) is_negative = true, n *= -1;

	while (n) { // Mod/Div to pull out digits, they'll be stored in reverse order
		values_in_reverse[stepper++] = n % base;
		n /= base;
	}
	if (is_negative) values_in_reverse[++stepper] = 37;
	values_in_reverse[stepper] = '\0'; // Stepper now conveniently stores len

	for (int i = stepper -1 , j = 0; j < stepper; --i, ++j) {
		sOut[j] = char_lookup[values_in_reverse[i]];
	}
	sOut[stepper] = '\0';
}

int codontoi(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;
}

void itocodon(char codon[4], int i) {
	char codon_table[4] = {'C', 'T', 'A', 'G'};
	for (int i = 0; i < 4; ++i) codon[i] = 'C';

	if (i >= 64) {
		printf("Value too large to store in single codon. Exiting");
		exit(1);
	}
```

```
        for (int x = 3; x >= 0; --x) {
                codon[x] = codon_table[i % 4];
                i /= 4;
        }
}

void add_in_base(char* s1, char* s2, char* sum, int base) {
        long x, y, z;

        x = my_atoi(s1, base);
        y = my_atoi(s2, base);
        z = x + y;
        my_itoa(z, sum, base);
}
```

Header File:

```
#ifndef H_BASE_CONVERSION
#define H_BASE_CONVERSION

int my_pow(int, int);
long my_atoi(char*, int);
void my_itoa(long, char*, int);
int codontoi(char*);
void itocodon(char*, int);
void add_in_base(char*, char*, char*, int);

const unsigned char MAX_INPUT_LENGTH = 255;
#endif
```