//

// main.c

// Chpt9PP2

//

// Created by Randy McMillan on 11/3/13.

// Copyright (c) 2013 Randy McMillan. All rights reserved.

//

/\*

\*

2. A palindrome consists of a word or deblanked, unpunctuated phrase that is spelled exactly the same when the letters are reversed. Write a recursive func- tion that returns a value of 1 if its string argument is a palindrome. Notice that in palindromes such as level, deed, sees, and Madam I’m Adam (madamimadam), the first letter matches the last, the second matches the next-to-last, and so on.

\*

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//#include <stdbool.h>

#define LINE 80

#define ZERO 0

#define BLANK ' '

char input[LINE];

char output[LINE];

int num = 0;

char \*reverse\_string(char \*str);

void myMain();

void promptInput();

int determinePalindrone(char \*string,int z);

int StringIsPalindrome(const char\* string);

void deBlank(char \*input, char \*output, int loopLength);

char \*scanLine(char \*dest, int dest\_len);

int main(int argc, const char \*argv[])

{

myMain();

return 0;

}

void myMain()

{

promptInput();

deBlank(scanLine(input, LINE), output, (int)strlen(input) + 1);

//printf("\n%s\n", output);

//printf("\n%i\n",determinePalindrone(output,(int)strlen(output)));

int boolean = determinePalindrone(output, (int)strlen(output));

if (boolean) {

printf("%s is a Palindrone when it is de blanked and evaluated as --> \n%s\n",input,output);

}else{

printf("%s is NOT a Palindrone when it is de blanked and evaluated as --> \n%s\n",input,output);

}

}

void promptInput()

{

printf("Please enter a string to be analyzed --> ");

}

void deBlank(char \*input, char \*output, int loopLength)

{

// mind my p's and q's

int p = ZERO;

int q = ZERO;

while (p < loopLength) {

if (input[p] != BLANK) { // if not blank

output[q] = input[p];

q++; // increment output index by one when true

}

p++;// traverse array

}

}

int determinePalindrone(char \*string,int z){

if (!strcmp(string, reverse\_string(string))) {

//if (strcmp(string, reverse\_string(string))) {

//I've tried multiples ways

/\*

What am I missing with this syntax?

the C reference states:

2.14.10 strcmp

Declaration:

int strcmp(const char \*str1, const char \*str2);

Compares the string pointed to by str1 to the string pointed to by str2.

Returns zero if str1 and str2 are equal. Returns less than zero or greater than zero if str1 is less than or greater than str2 respectively.

What am I missing with the syntax?

This has been happening for several chapters and I can't get logic statments to evaluate they way they should no matter what kindo of conditions I make.

\*/

//printf("Why doesnt this work?");

} else {

//printf("I can't get any logic to work in my programs. It's like there is some kind of bug in Xcode or something. It's effecting my work. ");

}

return StringIsPalindrome(string);

}

///common reverse string technique

char \*reverse\_string(char \*str)

{

char temp;

size\_t len = strlen(str) - 1;

size\_t i;

size\_t k = len;

for (i = 0; i < len; i++) {

temp = str[k];

str[k] = str[i];

str[i] = temp;

k--;

if (k == (len / 2)) {

break;

}

}

return str;

}

//<http://dystopiancode.blogspot.com/2012/07/palindrome-checking-algorithm-in-ansi-c.html>

int StringIsPalindrome(const char\* string)

{

long length = strlen(string);//get a string length and assign it to length

int i;//create a counter

int isPalindrome = 1;//like a boolean flag

for(i = 0; i<(length/2); i++)//for while i is half the length of length and increment by 1

{

if(string[i]!=string[length-i-1])//if string[i] doesnt equal string[length minus 1 minus 1]

{

isPalindrome = 0;//if each letter matches the isPalindrome is never set to zero

break;

}

}

return isPalindrome;//returns 1 if isPalindrome is not set to zero

}

// Based on Figure 8.15 in Book

char \*scanLine(char \*dest, int dest\_len)

{

int i, ch;

i = ZERO;

for (ch = getchar(); ch != '\n' && ch != EOF && i < dest\_len - 1; ch = getchar()) {

dest[i++] = ch;

}

dest[i] = '\0';

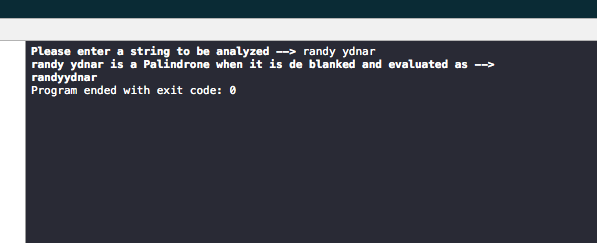
while (ch != '\n' && ch != EOF) {

ch = getchar();

}

return dest;

}



//

// main.c

// Chpt9PP3

//

// Created by Randy McMillan on 11/12/13.

// Copyright (c) 2013 Randy McMillan. All rights reserved.

//

/\*

3. Write and test a recursive function that returns the value of the following recursive definition:

f(x) = 0 if x <= 0

f(x) = f(x-1) + 2 otherwise

What set of numbers is generated by this definition?

\*/

#include <stdio.h>

double input;

double firstFunction(double);

double compositeFunction(double);

void promptInput();

void promptInput()

{

printf("Please enter a number --> ");

}

int main(int argc, const char \* argv[])

{

promptInput();

scanf("%lf",&input);

printf("\ny = %lf\n",firstFunction(input));

//I've tried this a couple different ways. I'm not sure what I'm missing.

main(argc,argv);

return 0;

}

double firstFunction(double x){

if (x<=0) {

x=0;

}else{

x = compositeFunction(x);

}

return x;

}

double compositeFunction(double x){

return (x-1)+2;

}

