/\* -----------------------------------------------

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: No. 3 Part 1

Purpose: Read a phrase and determine whether it is a PALINDROME

----------------------------------------------- \*/

#include <iostream>

#include "STACKPAC.h"

// function prototytpes

template<class T> T getInput();

template<class T> T reverseStack(T s);

template<class T> bool isStackEqual(T s1, T s2);

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

char cont = 'y'; // sentinel value to determine if the user wants to stop the program

while(cont == 'y')

{

STACK<char, 100> s1 = getInput<STACK<char, 100>>();

STACK<char, 100> s2 = reverseStack<STACK<char, 100>>(s1);

if (isStackEqual(s1, s2)) { std::cout << "This statement is a PALINDROME" << std::endl; }

else { std::cout << "This statement is NOT a PALINDROME" << std::endl; }

std::cout << "\tCONTINUE(y/n)? ";

std::cin >> cont;

cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cout << std::endl << std::endl;

}

system("pause");

return 0;

}

template<class T>

T getInput()

{

/\* name: getInput

input: N/A

output: STACK s

purpose: prompts and gets input from the user \*/

T s;

char c;

std::cout << "Enter a phrase: ";

while (c != '\n')

{

cin.get(c);

if (isalpha(c)) { s.push(tolower(c)); }

}

return s;

}

template<class T>

T reverseStack(T s)

{

/\* name: main

input: STACK s

output: STACK r

purpose: reverses the order of a stack by using another stack \*/

T r = s;

r.clear();

while (!s.isEmpty())

{

r.push(s.pop());

}

return r;

}

template<class T>

bool isStackEqual(T s1, T s2)

{

/\* name: isStackEqual

input: STACK s1, STACK s2

output: bool

purpose: determines if two stacks are equal\*/

while (!s1.isEmpty())

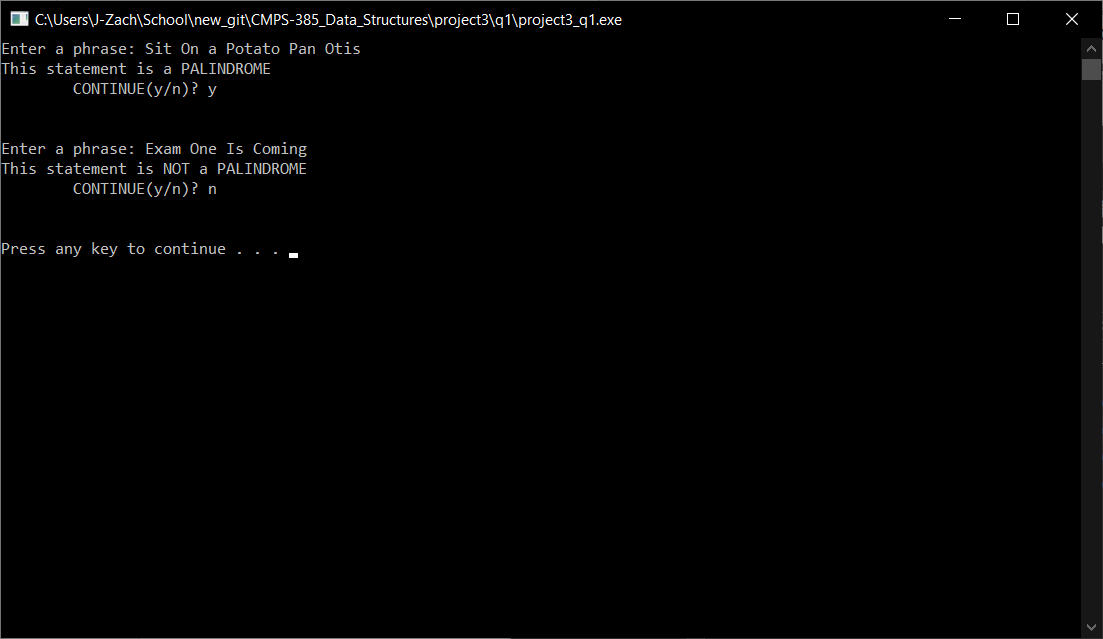
{

if (s1.pop() != s2.pop()) { return false; }

}

return true;

}



/\* -----------------------------------------------

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: No. 3 Part 2

Purpose: Given some arrays and class declarations, display each original

array followed by their sorted form

----------------------------------------------- \*/

#include <iostream>

#include <string>

#include <algorithm>

// given class declaration

template <class T, int n>

class TWO

{

private:

T x[n];

public:

void Copy(T t[]);

void Display();

void SortArray();

};

// class member definitions

template <class T, int n>

void TWO<T, n>::Copy(T t[])

{

/\* name: Copy

input: array t

output: N/A

purpose: copies array t into array x \*/

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

{

x[i] = t[i];

}

}

template <class T, int n>

void TWO<T, n>::Display()

{

/\* name: Display

input: N/A

output: N/A

purpose: displays array x \*/

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

{

std::cout << x[i] << '\t';

}

}

template <class T, int n>

void TWO<T, n>::SortArray()

{

/\* name: SortArray

input: N/A

output: N/A

purpose: sorts array x \*/

std::sort(x, x+n);

}

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

// given arrray definitions

int a[6] = {3, 9, 10, 7, 1, 8};

std::string Days[7] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};

float GPA[3] = {2.1, 4.2, 2.8};

TWO<int, 6> P;

P.Copy(a);

std::cout << "This is the original array a:\t";

P.Display();

P.SortArray();

std::cout << std::endl << "This is the sorted array a:\t";

P.Display();

std::cout << std::endl << std::endl;

TWO<std::string, 7> Q;

Q.Copy(Days);

std::cout << "This is the original array a:\t";

Q.Display();

Q.SortArray();

std::cout << std::endl << "This is the sorted array a:\t";

Q.Display();

std::cout << std::endl << std::endl;

TWO<float, 3> R;

R.Copy(GPA);

std::cout << "This is the original array a:\t";

R.Display();

R.SortArray();

std::cout << std::endl << "This is the sorted array a:\t";

R.Display();

std::cout << std::endl << std::endl;

system("pause");

return 0;

}

template<class T>

T getInput()

{

/\* name: getInput

input: N/A

output: STACK s

purpose: prompts and gets input from the user \*/

T s;

char c;

std::cout << "Enter a phrase: ";

while (c != '\n')

{

cin.get(c);

if (isalpha(c)) { s.push(tolower(c)); }

}

return s;

}

template<class T>

T reverseStack(T s)

{

/\* name: main

input: STACK s

output: STACK r

purpose: reverses the order of a stack by using another stack \*/

T r = s;

r.clear();

while (!s.isEmpty())

{

r.push(s.pop());

}

return r;

}

template<class T>

bool isStackEqual(T s1, T s2)

{

/\* name: isStackEqual

input: STACK s1, STACK s2

output: bool

purpose: determines if two stacks are equal\*/

while (!s1.isEmpty())

{

if (s1.pop() != s2.pop()) { return false; }

}

return true;

}

