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

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: No. 6 Part 1

Purpose: Write a program to read a phrase and determine wheter it is a palindrome

or not. Must use the pointer implementation of stack and queue.

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

#include <iostream>

#include <string>

class LinkedList

{

private:

struct node

{

char c;

node \*next;

};

node \*list;

public:

LinkedList() { list = NULL; }

void push(char x)

{

/\* name: push

input: char x

output: N/A

purpose: adds an element into the linked list stack \*/

node \*p = new(node);

p->c = x;

p->next = list;

list = p;

}

char pop()

{

/\* name: pop

input: N/A

output: int

purpose: removes and returns an element from the linked list stack \*/

node \*p = list;

int value = p->c;

list = p->next;

return value;

}

bool isEmpty()

{

/\* name: isEmpty

input: N/A

output: bool

purpose: determines if the linked list stack is empty \*/

return (list == NULL) ? true : false;

}

};

// function prototypes

std::string getInput();

LinkedList makeStack(std::string input);

bool isPalindrome(LinkedList phrase);

bool getContinue();

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

do

{

std::string input = getInput();

LinkedList phrase = makeStack(input);

std::string output = (isPalindrome(phrase)) ? "That is a palindrome" : "That is not a palindrome";

std::cout << '\t' << output << std::endl;;

} while (getContinue());

system("pause");

return 0;

}

std::string getInput()

{

/\* name: getInput

input: N/A

output: string

purpose: gets input from the user \*/

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

std::string input;

getline(std::cin, input);

return input;

}

LinkedList makeStack(std::string input)

{

/\* name: makeStack

input: string input

output: LinkedList

purpose: converts user input into a stack \*/

LinkedList phrase;

for (char c : input)

{

if (isalpha(c))

{

phrase.push(tolower(c));

}

}

return phrase;

}

bool isPalindrome(LinkedList phrase)

{

/\* name: isPalindrome

input: LinkedList phrase

output: bool

purpose: determines if a phrase is a palindrome \*/

LinkedList temp = phrase;

LinkedList reverse;

while (!temp.isEmpty())

{

reverse.push(temp.pop());

}

while (!phrase.isEmpty())

{

if (phrase.pop() != reverse.pop())

{

return false;

}

}

return true;

}

bool getContinue()

{

/\* name: getContinue

input: N/A

output: bool

purpose: asks the user if they want to continue the program \*/

char cont;

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

std::cin >> cont;

std::cin.ignore(1);

std::cout << std::endl;

return (tolower(cont) == 'y') ? true : false;

}

