Kahlil Bello

Data Structures Proj 7

Oct 24/2018

**PROBLEM 1**

main.cpp

#include <iostream>

#include "STACKPAC.h"

using namespace std;

int main()

{

QUEUE\_LIST<char> queue; //q->[ |/]

STACK\_LIST<char> stack; //q->[ |/]

char c;

char again = 'y';

do

{

queue.clearq();

stack.clears();

cout << "Type in a sentence to test for palindrome: ";

//read input and ignore whitespace and specialchars

while (cin.get(c), c != '\n')

{

if (isalpha(c))

{

c = toupper(c);

queue.pushq(c);

stack.pushs(c);

}//endif

}//endwhile

//check for palindrome

char c1, c2;

while (!queue.emptyq())

{

c1 = queue.popq();

c2 = stack.pops();

if (c1 != c2) { break; }//break out of loop if not equal

}//endwhile

if (queue.emptyq())

{

cout << "You have a palindrome\n";

}//endif

else

{

cout << "you do not have a palindrome\n";

}//endelse

cout << "Continue? (y/n). . . ";

cin >> again;

cin.ignore(1, '\n');

} while (again == 'Y' || again == 'y');//end dowhile

system("pause");

return 0;

}//end main

**HEADER FILE**

#pragma once

#ifndef STACKPAC

#define STACKPAC

#include <iostream>

template <class T>

class STACK\_LIST

{

private:

struct node

{

T info;

node \*next;

};//end struct

node \*stack;

public:

STACK\_LIST() { stack = NULL; }

//---------add node to front of list------

void pushs(T x)

{

node \*p;

p = new(node);

p->info = x;

p->next = NULL;

if (stack == NULL)

{

stack = p;

}//endif

else

{

p->next = stack;

stack = p;

}//endelse

}//end pushs

//----------test whether stack is empty--------

bool emptys()

{

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

}//end emptys

T pops()

{

T x;

node \*p;

p = stack;

x = stack->info;

stack = stack->next;

delete(p);

return x;

}//end pops

void clears()

{

stack = NULL;

}//end clears

};//end STACK\_LIST

template <class G>

class QUEUE\_LIST

{

private:

struct node

{

G info;

node \*next;

};//end struct node

node \*front, \*rear;

public:

QUEUE\_LIST() { front = NULL; rear = NULL; }

//------------add a node to the rear of the queue

void pushq(G x)

{

node \*p = new(node);

p->info = x;

p->next = NULL;

if (front == NULL)

{

front = p;

rear = p;

}//endif

else

{

rear->next = p;

rear = p;

}//endelse

}//end pushq

//--------test whether queue is empty

bool emptyq()

{

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

}

//----------pop the first node

G popq()

{

node \*p = front;

G x;

x = front->info;

front = front->next;

delete(p);

return x;

}//end popq

void clearq()

{

front = NULL;

}//end clearq

};//end QUEUE\_LIST

template <class F>

class OrderedLinkedList

{

private:

struct node

{

F info;

node \*next;

};//end struct node

node \*list;

public:

OrderedLinkedList() { list = NULL; }

void inser(F x)

{

node \*p = list;

node \*q = list;

node \*r;

//create a new node

r = new(node);

r->info = x;

r->next = NULL;

while (p != NULL && x > p->info)

{

q = p;

p = p->next;

}//endwhile

if (p == list) //x is the first info

{

list = r;

r->next = p;

}//endif

else if (p == NULL) //x is the last info

{

q->next = r;

}//endelseif

else //x is neither first nor last info

{

r->next = p;

q->next = r;

}

}//end insert

void display()

{

node \*p = list;

while (p != NULL)

{

std::cout << p->info << "->";

p = p->next;

}//endwhile

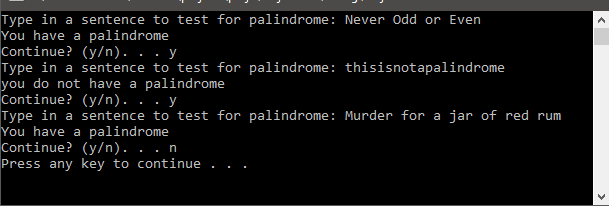
std::cout << "NULL\n";

}//end display

};//end OrderedLinkedList

#endif

OUTPUT:



**PROB 2**

#include <iostream>

#include <cmath>

using namespace std;

class POLY

{

private:

struct node

{

int coef;

int pwr;

node \*next;

};//end sruct node

node \*front, \*rear;

public:

POLY() { front = NULL; rear = NULL; }

void insert(int c, int pw)

{

node \*p;

p = new(node);

p->coef = c;

p->pwr = pw;

p->next = NULL;

if(front==NULL)

{

front = p;

rear = p;

}//endif

else

{

rear->next = p;

rear = p;

}//endelse

}//end pushq

bool emptyq()

{

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

}//end emptyq

void display()

{

node \*p = front;

while (p != NULL)

{

if (p->coef > 0)

{

cout << "+" << p->coef;

}//endif

else if(p->coef < 0 )

{

cout << "" << p->coef;

}//endelse

cout << "x^" << p->pwr;

p = p->next;

}//endwhile

}//end display

int value(int a)

{

int val = 0;

node \*p = front;

while (p != NULL)

{

val += (p->coef)\*pow(a, p->pwr);

p = p->next;

}//endwhile

return val; //return polynomial

}//end value

};//end class QUEUE\_LIST

int main()

{

POLY P;

int c, pw; //for coef and power

cout << "Enter the coefficient and power of a term (0,0 to stop): ";

cin >> c >> pw;

while ((c != 0) || (pw != 0))

{

P.insert(c, pw);

cout << "Enter the coefficient and power of a term (0,0 to stop): ";

cin >> c >> pw;

}//endwhile

//--------display poly-------

cout << "This is your function: ";

P.display();

cout << "\n\n";

int a;

cout << "Enter a value to evaluate the polynomial: ";

cin >> a;

cout << "F(" << a << ") = " << P.value(a) << endl;

system("pause");

return 0;

}//end main

**OUTPUT:**

