Lab6 23k0634

Q1

#include<iostream>

using namespace std;

// Implement and insert the values “BORROWROB” in the stack and identify if it’s palindrome

// or not. Use the push and pop functions to accomplish this (Note: Use Arrays to accomplish

// this).

int maxsize = 10;

class Stack

{

    string arr[10];

    int top;

    public:

    Stack(){top = -1;}

    bool isFull()

    {

        if (top == maxsize-1){

            return true;

        }

        else{

            return false;

        }

    }

    void push(string data)

    {

        if (isFull()){

            cout<<"Can not push. limit reached"<<endl;

        }

        else{

            top++;

            arr[top] = data;

        }

    }

    bool isEmpty()

    {

        if (top == -1){

            return true;

        }

        else{

            return false;

        }

    }

    string pop()

    {

        string temp;

        if (isEmpty()){

            cout<<"Can not pop, stack empty."<<endl;

        }

        else{

            temp = arr[top];

            top--;

        }

        return temp;

    }

    void size()

    {

        cout<<top<<endl;

    }

    bool isPalindrome()

    {

        string upperHalf[4];

        string lowerHalf[4];

        int temp = top;

        int i=0;

        while(temp > 4){

            upperHalf[i] = pop();

            i++;

            temp--;

        }

        string middle = pop();

        temp--;

        i=0;

        while(temp > -1){

            lowerHalf[i] = pop();

            temp--;

            i++;

        }

        // for (int i=0;i<4;i++){

        //     cout<<upperHalf[i]<<" ";

        // }cout<<endl;

        // for (int i=0;i<4;i++){

        //     cout<<lowerHalf[i]<<" ";

        // }

        bool palindrome = true;

        int j=3;

        for (int i=0;i<4;i++){

            if (lowerHalf[i] != upperHalf[j]){

                palindrome = false;

                cout<<"The stack is NOT a palindrome!";

                return palindrome;

            }

            j--;

        }

        cout<<"The stack is a palindrome!";

    }

};

int main()

{

    Stack stack;

    // test for palindrome

    // stack.push("B");

    // stack.push("O");

    // stack.push("R");

    // stack.push("R");

    // stack.push("O");

    // stack.push("R");

    // stack.push("R");

    // stack.push("O");

    // stack.push("B");

    // test for not a palindrome (original question)

    // stack.push("B");

    // stack.push("O");

    // stack.push("R");

    // stack.push("R");

    // stack.push("O");

    // stack.push("W");

    // stack.push("R");

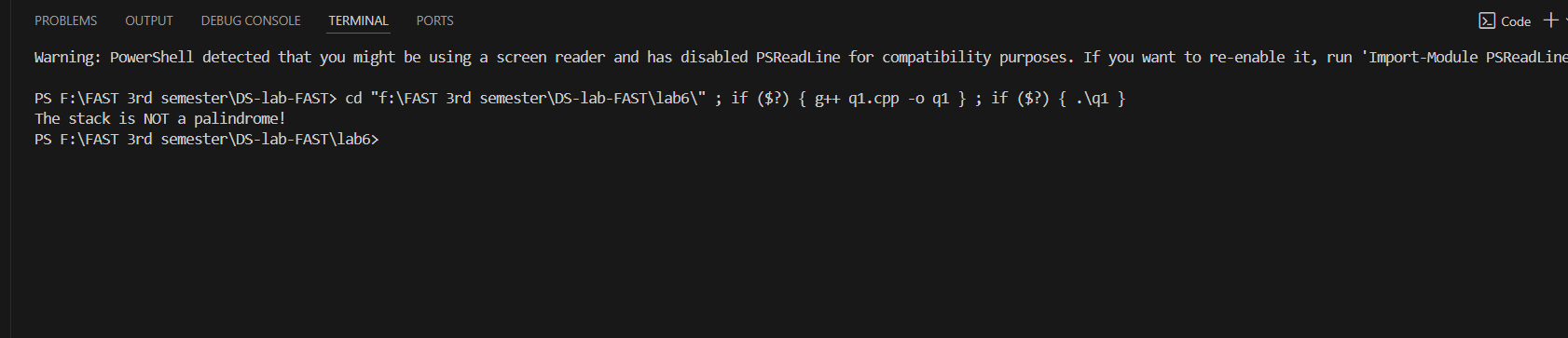
    // stack.push("O");

    // stack.push("B");

    stack.isPalindrome();

    return 0;

}



Q2

#include<iostream>

using namespace std;

// Implement a Queue based approach where assume you are the cashier in a supermarket and

// you need to make checkouts. Customer ID’s Are 13,7,4,1,6,8,10. (Note: Use Arrays to

// accomplish this task with enqueue and dequeue).

int Qsize = 10;

class Queue

{

    int arr[10];

    int front;

    int back;

    public:

    Queue(){front=-1;back=-1;}

    void enqueue(int data)

    {

        if (back < Qsize){

            back++;

            arr[back] = data;

        }

        else{

            cout<<"Queue Full"<<endl;

        }

    }

    void dequeue()

    {

        if (front < back){

            front++;

            cout<<arr[front]<<endl;

        }

        else{

            cout<<"Queue empty"<<endl;

        }

    }

    void print()

    {

        for (int i=front+1;i<=back;i++){

            cout<<arr[i]<<" ";

        }

        cout<<endl;

    }

};

int main()

{

    Queue que;

    //enqueueing

    que.enqueue(13);

    que.enqueue(7);

    que.enqueue(4);

    que.enqueue(1);

    que.enqueue(6);

    que.enqueue(8);

    que.enqueue(10);

    que.print();

    //dequeueuing

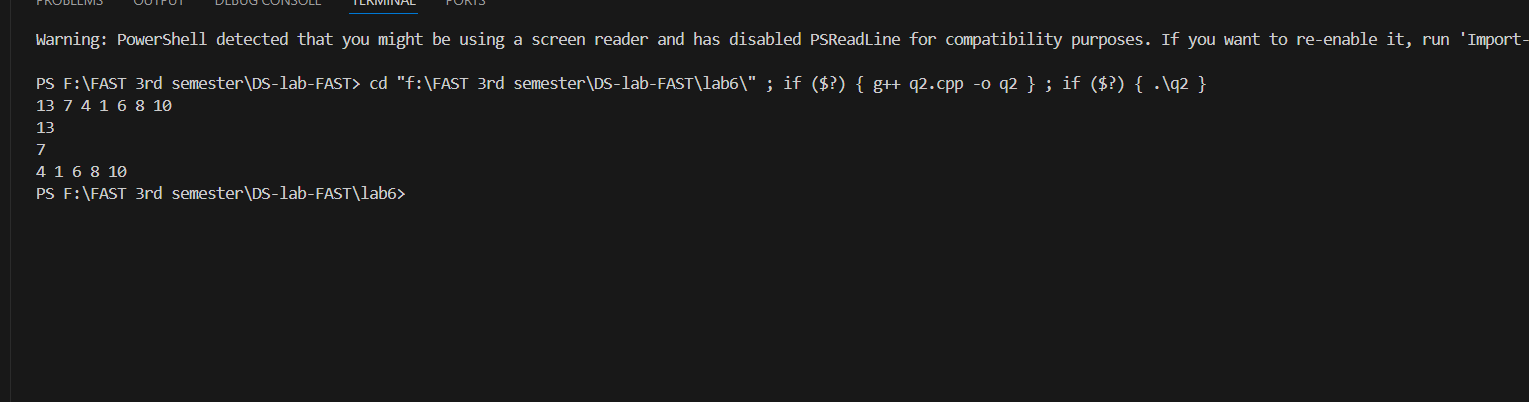
    que.dequeue();

    que.dequeue();

    que.print();

    return 0;

}



Q3

#include <iostream>

#include<string>

using namespace std;

class Node {

    public:

        string data;

        Node\* next;

};

class LinkedListStack {

    private:

        Node\* top;

    public:

        LinkedListStack() : top(nullptr) {}

        void push(string val){

            Node\* newNode = new Node();

            newNode->data = val;

            newNode->next = top;

            top = newNode;

        }

        string pop() {

            if(!isEmpty()){

                Node\* temp = top;

                string val = top->data;

                top = top->next;

                delete temp;

                return val;

            }

            return "";

        }

        string peek() {

            if(!isEmpty()){

                return top->data;

            }

            return "";

        }

        bool isEmpty(){

            return top == nullptr;

        }

        void display() {

            Node\* current = top;

            while(current != nullptr){

                cout << current->data << " ";

                current = current->next;

            }

            cout << endl;

        }

};

double evaluateExpression() {

    double result = 12;

    result += 13;

    result -= 5 \* (0.5 + 0.5);

    result += 1;

    return result;

}

int main() {

    LinkedListStack stack;

    stack.push("x");

    stack.push("=");

    stack.push("12");

    stack.push("+");

    stack.push("13");

    stack.push("-");

    stack.push("5");

    stack.push("\*");

    stack.push("(");

    stack.push("0.5");

    stack.push("+");

    stack.push("0.5");

    stack.push(")");

    stack.push("+");

    stack.push("1");

    cout<<endl<< "Stack before evaluation:"<<endl<<endl;

    stack.display();

    int result = evaluateExpression();

    stack.push(to\_string(result));

    cout<<endl<<"Stack after evaluation (result on top):"<<endl<<endl;

    stack.display();

    cout<<endl;

    return 0;

}

