Muzamil

P20-0108

Stack with Array

Task#1

```cpp
#include<iostream>
#define n 9
using namespace std;
class stack{
    int top;
    public:
        int arr[n];
        stack(){
            top=-1;
        }

        bool push(int data){
            if(top>n-1){
                //cout<<"staack is overflowing ";
                return false;
            }
            else{
                top++;
                arr[top]=data;
                return true;
            }
        }
        int peek(){
            if(top>n-1){
                cout<<"stack overflow ";
            }
            else{
                cout<<arr[top]<<" ";

            }
```

```
int pop(){
    if(top<0){
        cout<<"Stack is empty ";
        return 0;
    }
    else{
        --top;
        int data=arr[top];
        return data;
    }
}
bool isfull(){
return(top>n);
}
bool isempty(){
    return(top<0);
}

int main(){
    stack s1;


    s1.push(12);
    s1.push(24);
    s1.push(1);
    s1.push(2);
    s1.push(3);
    s1.push(8);
    s1.push(7);
    s1.push(6);
    s1.push(9);
//  s1.push(90);
//  s1.push(7);
//  s1.push(6);
    s1.push(9);
    s1.push(90);
//  s1.isfull();
    for(int i=0;i<11;i++){
        s1.peek();
        s1.pop();
    }

}
```

Output

```
stack overflow 9 6 7 8 3 2 1 24 12 -1 Stack is empty
--------------------------------
Process exited after 0.0808 seconds with return value 0
```

Task#2

Stack With Linked list

```
#include<iostream>
using namespace std;

class node{
    int top;
    public:

        node *next;
        int data;
        node(){
            top=0;
            this->next=NULL;
            data=0;
        }

};
```

```
class linked{
    public:
        node *top;
        linked(){
            this->top=NULL;
        }
        bool push(int n){
            node *tmp;
            tmp=new node;
            tmp->data=n;
            if(top==NULL){
                top=tmp;

            }
            else{
                tmp->next=top;
                top=tmp;


            }
        }
```

```
int pop(){
    node *tmp;
    tmp=top;
    if(top==NULL){
        cout<<"stack is empty ";
    }
    else{
        top=top->next;
        delete top;
    }
}
```

```
 int peek(){
     node *tmp;
     tmp=top;
     if(top==NULL){
         cout<<"Stack is empty ";
     }
     else{
         cout<<"top element is  "<<top->data<<" ";

     }
 }
```

```
void dislay(){
    node *tmp;
    tmp=top;
    while(tmp!=NULL){
        cout<<tmp->data<<" ";
        tmp=tmp->next;
    }
}

void reverse()
{

    node* current = top;
    node *prev = NULL, *next = NULL;

    while (current != NULL) {
        next = current->next;

        current->next = prev;

        prev = current;
        current = next;
    }
    top = prev;
}
```

```
int main(){
    linked l;
    l.push(12);
    l.push(1);
        l.push(2);
    l.push(11);
        l.push(3);
    l.push(15);
        l.push(155);
    l.push(17);
        l.push(162);
    l.push(113);


    cout<<"before reversing the elements are "<<endl;

    l.dislay();
    cout<<endl;
    cout<<"after reversing the elements are "<<endl;
    l.reverse();
    l.dislay();
    cout<<endl;
    cout<<endl;
```

```
 l.pop();
 //l.peek();
 cout<<endl;
 l.peek();
 l.pop();
 //l.pop();l.pop();l.pop();l.pop()
 l.peek();
 l.peek();
```

OutPut

```
before reversing the elements are
113 162 17 155 15 3 11 2 1 12
after reversing the elements are
12 1 2 11 3 15 155 17 162 113
```

Task#3

```
using namespace std;

int prec(char c) {
    if(c == '^')
        return 3;
    else if(c == '/' || c=='*')
        return 2;
    else if(c == '+' || c == '-')
        return 1;
    else
        return -1;
}
```

```
void infixToPostfix(string s) {

    stack<char> st;
    string result;

    for(int i = 0; i < s.length(); i++) {
        char c = s[i];


        if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9'))
            result += c;


        else if(c == '(')
            st.push('(');


        else if(c == ')') {
            while(st.top() != '(')
            {
                result += st.top();
                st.pop();
            }
            st.pop();
        }
```

```
        else {
            while(!st.empty() && prec(s[i]) <= prec(st.top())) {
                result += st.top();
                st.pop();
            }
            st.push(c);
        }
    }


    while(!st.empty()) {
        result += st.top();
        st.pop();
    }

    cout << result << endl;
}


int main() {
    string exp ;
    cin>>exp;
    infixToPostfix(exp);
    return 0;
}
```

Ouput

```
enter the string that you want to convert it from infix to postfix
hello+y-(a+b)/h
helloy+ab+h/-

--------------------------------
Process exited after 14.99 seconds with return value 0
Press any key to continue . . .
```

Task#4

Queue With Array

```
#include<iostream>
#define n 4
using namespace std;

class Queue {

int front, rear, size;
public:
    int arr[n];
    Queue(){
        front=-1;
        rear=-1;
    }
    bool enqueue(int data){
        if(rear==n-1){
            cout<<"Overflow ";
            return 0;
        }

        else{
            rear++;
            arr[rear]=data;
        //  cout<<arr[rear];
            return true;
        }
    }
    int dequeue(){
```

```
int dequeue(){
    if(front==rear){
        cout<<"empty  ";
    }
    else{
//                      front++;
//                          //      front++;
//
        cout<<"deleted element is "<<arr[front]<<endl;
                          //  front++;

        front++;
        int s=arr[front];
        cout<<"Now front is Pointing to "<<s<<endl;

        return s;
    }
}
```

```
int peek(){
    if(rear<0){
        cout<<"underflow ";
    }
    else{
        //cout<<arr[rear];
        int p=arr[rear];
        cout<<"Top element is "<<p<<endl;
    }
}
```

```
bool isfull(){
    if(rear==n){
        cout<<"oops Queue is Full ";
        return true;
    }
    else{
        return false;
    }
}
```

```
int main(){
    Queue q;
    q.enqueue(2);
    q.enqueue(23);
    q.enqueue(245);
    q.isfull();
    q.enqueue(237);
    q.isfull();

    q.peek();
q.dequeue();

//q.peek();
}
```

Output

```
Top element is 237
deleted element is 32
Now front is Pointing to 2

--------------------------------
```

Task#5

Queue With Linked List

```cpp
#include<iostream>
#define CAPACITY 100
using namespace std;
class node{
    public:
        node *next;
        int data;
        node(){
            this->next=NULL;
            data=0;


        }


};
class Queue{
    public:
        node *front;
        node *rear;
        int size=0;
        Queue(){

        front=0;
        rear=0;

        }
 void enqueue(int n){
     node *tmp;
     tmp=new node;
     tmp->data=n;
     if(front==0 && rear==0){
         front=tmp;
         rear=tmp;
     }
     else{
         rear->next=tmp;
         rear=tmp;

     }
     size++;
 }
```

```
void display(){
    node *tmp;
    if(front==0&&rear==0){
        cout<<"Queue is empty ";
    }
    else{
        tmp=front;
        while(tmp!=NULL){
            cout<<tmp->data<<" ";
            tmp=tmp->next;
        }
    }
}
```

```
void dequeue(){
    node *tmp;
    tmp=front;
    if(front==0 && rear==0){
        cout<<"QUEUE is alread empty "<<endl;
    }
    else{


    tmp=front;
    front=front->next;
    cout<<"deleted data is "<<tmp->data<<" "<<endl;

    delete tmp;}
}
```

```
    int isFull(){
    return (size > CAPACITY);
}
```

```
int main(){
    Queue Q;
    Q.enqueue(2);
    Q.enqueue(1);
    Q.display();
    cout<<endl;
    Q.dequeue();
//  Q.display();
}
```

Output

```
2 1
deleted data is 2

--------------------------------
```