LAB 06

TASK 01:

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

template<typename T>
class Stack{
    private:
        T* stackArray;
        int stackTop;
    public:
        Stack() : stackArray(nullptr), stackTop(-1) {}

        void push(T data){
            stackArray=insertValue(stackArray,stackTop+1,data);
            stackTop++;
        }
        void pop(void){
            if(stackTop<0) return;
            stackArray=removeValue(stackArray,stackTop+1);
            stackTop--;
        }
        T top(void){
            if(stackTop<0) return T();
            return stackArray[stackTop];
        }
        int size(void){
            return stackTop+1;
        }
        bool empty(){
            return stackTop<0;
        }
        T* insertValue(T* array,int size,T value){
            T* temp = new T[size+1];
            for(int i=0 ; i<size ; i++){
                temp[i]=array[i];
            }
            temp[size]=value;
            delete[] array;
            return temp;
        }


        T* removeValue(T* array,int size){
            T* temp = new T[size-1];
            for(int i=0 ; i<size-1 ; i++){
                temp[i]=array[i];
```

```
            }
            delete[] array;
            return temp;
        }



        void display(void){
            int curr=stackTop;
            while(curr>-1){
                cout<<stackArray[curr--]<<endl;
            }
        }

        Stack& operator=(const Stack& other) {
            if (this == &other) return *this;

            delete[] stackArray;
            stackTop = other.stackTop;

            if(other.stackArray!=nullptr){
                stackArray=new T[stackTop+1];
                for(int i=0; i<stackTop+1; i++) {
                    stackArray[i]=other.stackArray[i];
                }
            } else{
                stackArray=nullptr;
            }
            return *this;
        }

        Stack<T> reverseStack(){

        }

        ~Stack() {
            delete[] stackArray;
        }
};

template<typename T>
void insertAtBottom(Stack<T>& st, T value){
    if (st.empty()) {
        st.push(value);
        return;
    }

    T curr=st.top();
```

```
    st.pop();
    insertAtBottom(st, value);
    st.push(curr);
}

template<typename T>
void reverse(Stack<T>& st){
    if (st.empty()) return;

    T curr=st.top();
    st.pop();
    reverse(st);
    insertAtBottom(st,curr);
}



template<typename T>
bool checkPalindrome(Stack<T> stack){
    Stack<T> orignal=stack;
    Stack<T> temp=stack;

    reverse(temp);
    while(!orignal.empty()){
        if(orignal.top()!=temp.top()) return false;
        orignal.pop();
        temp.pop();
    };
    return true;
}

int main(){

    Stack<char> st;
    st.push('B');
    st.push('O');
    st.push('R');
    st.push('R');
    st.push('O');
    st.push('W');
    st.push('R');
    st.push('O');
    st.push('B');

    st.display();
    cout<<endl;
    if(checkPalindrome(st)) cout<<"PALINDROME";
    else cout<<"NOT A PALINDROME";
```

```
    return 0;
}
```

```
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> g++ TASK01.cpp
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> ./a.exe
B
O
R
W
O
R
R
O
B

NOT A PALINDROME
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06>
```

TASK 02

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

template<typename T>
class Queue{
    private:
        T* queueArray;
        int _rear;
    public:
        Queue() : queueArray(nullptr), _rear(-1) {}

        void enqueue(T data){
            queueArray=insertValue(queueArray,_rear+1,data);
            _rear++;
        }
        void dequeue(void){
            if(_rear<0) return;
            queueArray=removeValue(queueArray,_rear+1);
            _rear--;
        }
        T rear(void){
            if(_rear<0) return T();
            return queueArray[_rear];
        }

        T front(void){
```

```
        if(_rear<0) return T();
        return queueArray[0];
    }
    int size(void){
        return _rear+1;
    }
    bool empty(){
        return _rear<0;
    }
    T* insertValue(T* array,int size,T value){
        T* temp = new T[size+1];
        for(int i=0 ; i<size ; i++){
            temp[i]=array[i];
        }
        temp[size]=value;
        delete[] array;
        return temp;
    }


    T* removeValue(T* array,int size){
        T* temp = new T[size-1];
        for(int i=1 ; i<size ; i++){
            temp[i-1]=array[i];
        }
        delete[] array;
        return temp;
    }



    void display(void){
        for(int i=0 ; i<_rear+1 ; i++){
            cout<<queueArray[i]<<" ";
        }
    }

    Queue& operator=(const Queue& other) {
        if (this == &other) return *this;

        delete[] queueArray;
        _rear = other._rear;

        if(other.queueArray!=nullptr){
            queueArray=new T[_rear+1];
            for(int i=0; i<_rear+1; i++) {
                queueArray[i]=other.queueArray[i];
            }
```

```
            } else{
                queueArray=nullptr;
            }
            return *this;
        }

        ~Queue() {
            delete[] queueArray;
        }
};

class CustomerLane{
    private:
        Queue<int> customerLane;
    public:
        cutomerLane(){};

        void checkIn(int ID){
            customerLane.enqueue(ID);
        }

        void checkOut(void){
            customerLane.dequeue();
       }

       bool anyCustomer(void){
            return !customerLane.empty();
       }

       int customerBeingProcessed(void){
            return customerLane.front();
       }

};

int main(){
    CustomerLane lane;
    lane.checkIn(10);
    lane.checkIn(8);
    lane.checkIn(6);
    lane.checkIn(1);
    lane.checkIn(4);
    lane.checkIn(7);
    lane.checkIn(13);

    while(lane.anyCustomer()){
        char option;
        cout<<"CUSTOMER CHECKOUT (y/n):";
```

```
        cin>>option;

        if(option=='y'){
            cout<<endl<<"Customer "<<lane.customerBeingProcessed()<<" Checked
Out"<<endl;
            lane.checkOut();

        }
    }




    return 0;
}
```

```
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> g++ TASK02.cpp
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> ./a.exe
CUSTOMER CHECKOUT (y/n):y

Customer 10 Checked Out
CUSTOMER CHECKOUT (y/n):n
CUSTOMER CHECKOUT (y/n):y

Customer 8 Checked Out
CUSTOMER CHECKOUT (y/n):y

Customer 6 Checked Out
CUSTOMER CHECKOUT (y/n):y

Customer 1 Checked Out
CUSTOMER CHECKOUT (y/n):y

Customer 4 Checked Out
CUSTOMER CHECKOUT (y/n):y

Customer 7 Checked Out
CUSTOMER CHECKOUT (y/n):y

Customer 13 Checked Out
```

TASK 03

```
# include <iostream>
# include <string>
# include <math.h>


using namespace std;


template<typename T>
class Node{
    public:
        Node<T>* next;
        T data;
        Node(T value):data(value),next(nullptr) {};
};


template<typename T>
class Stack{
    private:
        Node<T>* head;
        int _size;
    public:
        Stack() : head(nullptr), _size(0) {}
        Stack(const Stack& other) : head(nullptr), _size(0) {
        Node<T>* curr=other.head;
        while (curr){
            push(curr->data);
            curr=curr->next;
        }
    }

        void push(T data){
            Node<T>* newNode=new Node<T>(data);
            newNode->next=head;
            head=newNode;
            _size++;
        }

        T pop(void){
            if(!head) return T();
            _size--;
            Node<T>* tmp=head;
            head=head->next;
            T val=tmp->data;
            delete tmp;
            return val;
```

```
        }

        T top(void){
            if(!head) return T();
            return head->data;
        }

        int size(void){
            return _size;
        }

        bool empty(){
            return head==nullptr;
        }

        void display(void){
            Node<T>* curr=head;
            while(curr){
                cout<<curr->data<<endl;
                curr=curr->next;
            }
        }

        Stack& operator=(const Stack& other) {
            if (this==&other) return *this;

            while (!empty()) pop();

            _size=other._size;
            Node<T>* curr=other.head;
            while(curr){
                push(curr->data);
                curr=curr->next;
            }
            return *this;
        }
    ~Stack() {
        while (!empty()) {
            pop();
        }
    }

};


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

```
    else return -1;
}

bool isOperator(char character){
    return (character=='+' || character=='-' || character=='*' ||
character=='/' || character=='^');
}

bool isAlphabet(char character){
    return ((character>='A' && character<='Z') || (character>='a' &&
character<='z'));
}

string infixToPostfix(string expression){
    Stack<char> operatorStack;
    string postfix=" ";
    int stringSize=expression.length();
    int index=1;
    for(int i=0 ; i<stringSize ; i++){
        char character=expression[i];
        char characterBefore=(i-1<0) ? ' ' :  expression[i-1];
        if(isspace(character) || isAlphabet(character) || character=='='){
            continue;
        } else if(character=='.'){
            postfix[index-1]=character;
        } else if((isalnum(character) && isalnum(characterBefore))){
            postfix[index-1]=character;
            postfix+=" ";
            index++;
        }else if(isalnum(character)){
            postfix+=character;
            postfix+=" ";
            index+=2;
        }else if(character=='('){
            operatorStack.push(character);
        }else if(character==')'){
            while(!operatorStack.empty() && operatorStack.top()!='('){
                postfix+=operatorStack.pop();
                postfix+=" ";
                index+=2;

            }
            operatorStack.pop();
        } else if(isOperator(character)){
            while(!operatorStack.empty() &&
precedence(character)<=precedence(operatorStack.top())){
                postfix+=operatorStack.pop();
                postfix+=" ";
```

```
                index+=2;
            }
            operatorStack.push(character);
        }
    }
    while(!operatorStack.empty()){
        postfix+=operatorStack.pop();
        postfix+=" ";
        index+=2;
    }
    return postfix;
}

int read(string expression, int index){
    while(expression[++index]!=' ') {};
    return index+1;
}
string getSubstring(string str, int startIndex, int endIndex) {
    if (startIndex<0) startIndex=0;
    if (endIndex>str.length()) endIndex=str.length();
    if (startIndex>endIndex) return "";

    return str.substr(startIndex, endIndex-startIndex);;
}

float convert(string num){
    float numConverted;
    try{
        numConverted=stof(num);
    } catch(...){
        numConverted=float(stoi(num));
    }

    return numConverted;
}

float solve(float num01,float num02, char operation){
    if(operation=='+') return num01+num02;
    else if(operation=='-') return num01-num02;
    else if(operation=='*') return num01*num02;
    else if(operation=='/') return num01/num02;
    else if(operation=='^') return pow(num01,num02);
}


float evaluatePostfix(string postfix){
    Stack<float> operandStack;
    int leftIndex=1;
    int stringSize=postfix.length();
```

```
    while(leftIndex<stringSize){
        int rightIndex=read(postfix,leftIndex);
        string data=getSubstring(postfix,leftIndex,rightIndex);
        if(data=="" && data==" "){
            continue;
        } else if(!isOperator(*data.c_str())){
            operandStack.push(convert(data));
        } else if(!operandStack.empty()){
            float num02=operandStack.pop();
            float num01=operandStack.pop();
            operandStack.push(solve(num01,num02,*data.c_str()));
            // cout<<num01<<" "<<*data.c_str()<<" "<<num02<<"
"<<operandStack.top()<<endl;
        }
        leftIndex=rightIndex;
    }
    return operandStack.top();
}


int main(){
    string equation="x=12+13-5*(0.5+0.5)+1";
    string postfix=infixToPostfix(equation);


    Stack<string> expressionStack;
    expressionStack.push(equation);
    expressionStack.push("x="+to_string((int)evaluatePostfix(postfix)));

    expressionStack.display();
    return 0;
}
```

```
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> g++ TASK03.cpp
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06> ./a.exe
x=21
x=12+13-5*(0.5+0.5)+1
PS C:\Users\phoni\OneDrive\Desktop\DS LAB\DS LAB 06>
```