Stack

Parenthesis, Postfix calculator , Infix to postFix conversion

By

Tahmid

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

int precedence(char c){
    if(c=='+' ||c=='-')return 1;
    else if(c=='*' || c=='/')return 2;
    else if(c=='^')return 3;
    return 0;
}

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

string infix_to_postfix(string s){
    string postfix="";
    stack<char> A;
    for(int i=0;i<s.length();i++){
        char c=s[i];
        if(!is_operator(c)){
            postfix+=c;
        }
        else {
        if(!A.empty()&& precedence(A.top())>precedence(c)){
            postfix+=A.top();
            A.pop();
        }
        A.push(c);
    }
    }
    while(!A.empty()){
        postfix+=A.top();
        A.pop();
    }
}
```

```
bool is_operator(char c){
    return c=='+' or c=='-' or c=='/' or c=='*';
}

int postfixCount(string a){
    stack<int> A;
    int ans=0;
    for(char c: a){
        if(!is_operator(c)){
            A.push(c-'0');
        }
        else if(c=='+' && !A.empty()){
            int a=A.top();
            A.pop();
            int b;
            if(!A.empty())  b=A.top();
            A.pop();
            A.push(a+b);
        }
        else if(c=='-' && !A.empty()){
            int a=A.top();
            A.pop();
            int b;
            if(!A.empty())  b=A.top();
            A.pop();
            A.push(b-a);
        }
        else if(c=='*' && !A.empty()){
            int a=A.top();
            A.pop();
            int b;
            if(!A.empty())  b=A.top();
            A.pop();
            A.push(a*b);
        }
        else if(c=='/' && !A.empty()){
            int a=A.top();
            A.pop();
            int b;
            if(!A.empty())  b=A.top();
            A.pop();
            A.push(b/a);
        }

    }
    ans=A.top();
    return ans;
}
bool isValid(string a){
    stack<char> A;
    for(char c: a){
        if(c=='{' or c=='[' or c=='('){
            A.push(c);
        }
        else if(c==']'){
            if(A.empty() or A.top()!='[') return false;
            A.pop();
        }
        else if(c=='}'){
            if(A.empty() or A.top()!='{') return false;
            A.pop();
        }
        else if(c==')'){
            if(A.empty() or A.top()!='(') return false;
            A.pop();
        }

    }
    return A.empty();
}
```

Reverse queue

```

void reverseQueue( queue<int>& A){
    if(A.empty()) return;
    int front=A.front();
    A.pop();
    reverseQueue(A);
    A.push(front);
}
```