```cpp
//Largest Rectangle in Histogram : This Problem will cover NGE and NSE(Next Smallest Ele) concept
#include<bits/stdc++.h>
using namespace std;
int main(){
    vector<int> height = {2,1,5,6,2,3};
    // {1,6,4,4,6,6}
    // {-1,-1,1,2,1,4}
    int n = height.size();
    vector<int> right(n,n);
    vector<int> left(n,-1);
    stack<int> st;
    int ans = -1;
    //find NSE in right
    for(int i=0; i<n; i++){
        while(!st.empty() and height[i]<height[st.top()]){
            right[st.top()] = i;
            st.pop();
        }
        st.push(i);
    }

    while(!st.empty()) st.pop();

    //find NSE in left
    for(int i=n-1; i>=0; i--){
        while(!st.empty() and height[i]<height[st.top()]){
            left[st.top()] = i;
            st.pop();
        }
        st.push(i);
    }
    //calculate largest rectangle
    for(int i=0; i<n; i++){
        ans = max(ans,height[i]*(right[i]-left[i]-1));
    }
    cout<<"Ans: "<<ans;
    return 0;
}
```

```cpp
//Evaluation of postfix evaluation
int eval(int v1,int v2, char op){
    if(op == '+'){
        return v1+v2;
    }
    else if(op == '-'){
        return v1-v2;
    }
    else if(op == '*'){
```

```
            return v1*v2;
        }
        else if(op == '^'){
            return v1^v2;
        }
        else{
            return v1/v2;
        }
    }

    int evaluatePostfix(string str){
        stack<int> st;
        for(int i=0; i<str.size(); i++){
            if(isdigit(str[i])){
                st.push(str[i]-'0');
            }
            else{
                int v2 = st.top();
                st.pop();
                int v1 = st.top();
                st.pop();
                st.push(eval(v1,v2,str[i]));
            }
        }
        return st.top();
    }

    int main(){
        string exp = "231*+9-";
        cout<<"Ans: "<<evaluatePostfix(exp);
        return 0;
    }




    //Evaluation of infix exp
    int precedence(char op){
        if(op=='^'){
            return 3;
        }
        if(op=='+' or op=='-'){
            return 1;
        }
        if(op=='*' or op=='/'){
            return 2;
        }
        return -1;
    }

    int evaluateInfix(string str){
        stack<int> digit;
        stack<char> ops;
```

```
        for(int i=0; i<str.size(); i++){
            if(isdigit(str[i])){
                digit.push(str[i]-'0');
            }
            else if(str[i]=='('){
                ops.push(str[i]);
            }
            else if(str[i]==')'){
                while(!ops.empty() and ops.top()!='('){
                    char op = ops.top();
                    ops.pop();

                    int v2 = digit.top();
                    digit.pop();
                    int v1 = digit.top();
                    digit.pop();
                    digit.push(eval(v1,v2,op));
                }
                ops.pop();
            }
            else{
                while(!ops.empty() and precedence(ops.top())>=precedence(str[i])){
                    char op = ops.top();
                    ops.pop();

                    int v2 = digit.top();
                    digit.pop();
                    int v1 = digit.top();
                    digit.pop();
                    digit.push(eval(v1,v2,op));
                }
                ops.push(str[i]);
            }
        }

        while(!ops.empty()){
            char op = ops.top();
            ops.pop();

            int v2 = digit.top();
            digit.pop();
            int v1 = digit.top();
            digit.pop();
            digit.push(eval(v1,v2,op));
        }

        return digit.top();
    }

    int main(){
        string str = "2*(5*(3+6))/5-2";
        cout<<"Ans: "<<evaluateInfix(str);
        return 0;
    }
```