FREEMIND

Date

Page

Lecture - 54

(Stacks - 1)

⇒ Introduction

LIFO → Last in first out.

Some funct. used

- push → insert at top
- pop → remove from top
- is empty
- is full
- size
- Top

recursion
↳ call stack

(1) Overflow :- Jab defined capacity se jyada elements push karwane ki try karenge tab overflow wali condition hogi

(2) Underflow :- Jab eke empty stack mein se element remove karwane ke liye pop lagayenge tbh Underflow ki condition hogi.

→ Jab empty stack me top nikalenge to -1 print hega. i.e → Jab top() == -1 then either there is condition of Underflow or empty array.

CODE

Array Implementation of Stacks

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

class stack {
    int capacity;
    int *arr;
    int top;
public:
```

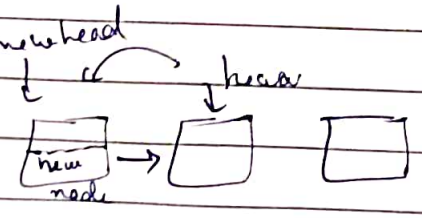
For Next page

```
stack(int c) {
    this -> capacity = c;
    arr = new int[c];
    this->top = -1;
}
Void push(int data) {
    if (this -> top == this-> capacity -1) {
        Cout << "overflow" \n";
        return;
    }
    this -> top++;
    this -> arr [this-> top] = data;
}
int pop() {
    if( this -> top == -1) {
        Cout << "Underflow\n";
        return INT_MIN;
    }
    this -> top--;
}
int getTop() {
    if( this-> top == -1) {
        Cout << "Underflow\n";
        return INT_MIN;
    }
    return this->arr(this-> top);
}
bool isEmpty() {
    return this->top == -1;
}
int size() {
    return this-> top +1;
}
```

```
bool isFull() {
    return this->top == this->capacity - 1;
}
};
int main() {
    Stack st(5);
    st.push(1);
    st.push(2);
    st.push(3);
    cout << st.getTop() << "\n";
    st.push(4);
    st.push(5);
    cout << st.getTop() << "\n";
    st.push(8);

    st.pop();
    st.pop();
    cout << st.getTop() << "\n";

    return 0;
}
```



Linked List implementation of Stacks

Head ko top banaker head se pehle ek ek element add karvate jao or head to usse (new node) pe shift karvate jao.

push → insert at head.

pop → remove from head.

Code

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

Class Node {
    public:
    Node * next;
    int data;
    Node (int d) {
        this → data = d;
        this → next = NULL;
    }
};

Class stack {
    Node* head;
    int capacity;
    int currsize;
    public:
    stack (int c){
        this → capacity = c;
        this → currsize = 0;
        head = NULL
    }
    bool isEmpty () {
        return this → head == NULL;
    }
    bool isFull () {
        return this → currsize == this → capacity;
    }
    Void push (int data){
        if (this → currsize == this → capacity) {
            cout << "overflow" << endl;
            return;
        }
```

```
    Node* new_node = new Node(data)
    new_node -> next = head;
    this -> head = new_node;
    this -> currsize++;
}
int pop() {
    if (this -> head == NULL) {
        cout << "Underflow\n";
        return;
    }
    Node * new_head = this -> head -> next;
    this -> head -> next = NULL;
    Node* toberemoved = this -> head;
    int result = ~~this -> head -> next~~ toberemoved -> data
    delete toberemoved
    this -> head = new_head
    return result;
}
int ~~size() {~~ getTop() {
    if (this -> head == NULL) {
        cout << "Underflow\n";
        return INT_MIN;
    }
    return this -> head -> data;
}
int size() {
    return currsize;
}
int main() {
    stack st(5);
    st.push(1) ..... (3)
    cout << getTop() << "\n";
    return 0;
}
```

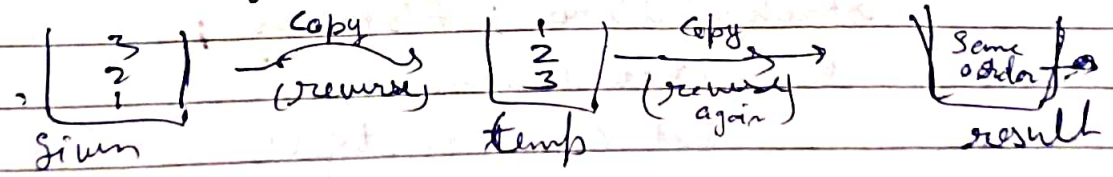
Inbuilt stack

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

int main() {
    stack<int> st;
//  st.pop();                          // fault
    st.push(1);
    st.push(2);
    st.push(3);
    cout << st.top() << "\n";          // 3
    st.push(4);
    st.push(6);
    cout << st.top() << "\n";          // 6
    st.pop();
    st.pop();
    cout << st.top() << "\n";          // 3
    cout << st.empty() << "\n";
    return 0;
}
```

Q→ Copy Stack:

Copy contents of one stack to another in same order.




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

```
stack<int> copystack(stack<int> &input) {

    stack<int> temp;
    while (not input.empty()) {
    //do the process till the time input stack
        doesn't become empty.
    int curr = input.top();
    input.pop();
    temp.push(curr);
    }
    stack<int> result;
    while (not temp.empty()) {
    int curr = temp.top();
        temp.pop();
        result.push(curr);
    }
    return result;
}
int main() {
    stack<int> st;
    st.push(1);
    st.push(2);
    st.push(3);
// stack<int> res = copystack(st);
    stack<int> res; f(st, res);          //For recursion
    while (not res.empty()) {
        int curr = res.top();
        res.pop();
        cout << curr << "\n";
    }
    return 0;
}
```

Recursion & Call Stack

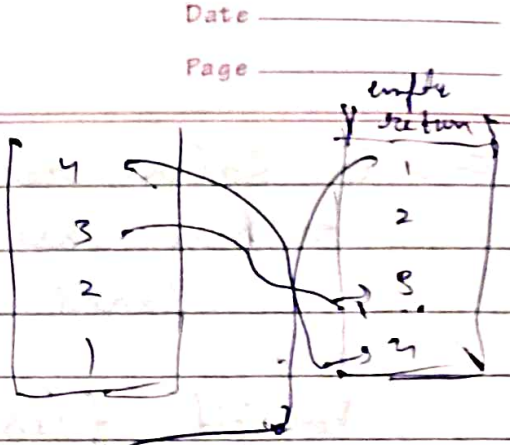
FREEMIND

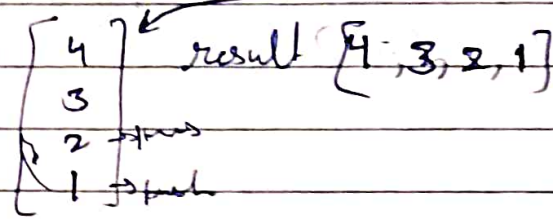
Date

Page

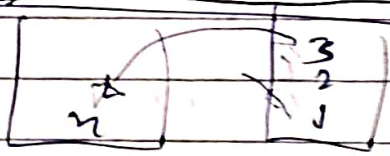
Copying Recursively

```
Void f(st, result) {                    stack<int> &
    if(st.empty()) {
    return;
    }
int curr = st.top()
    st.pop();
    f(st, result);
    result.push(curr);
}
```





Q2) Insert at bottom !



1) Pehle saara ka saara data kisi or stack (2nd) me daalda.
2) Phir vo element uss (1st) empty stack me daalda.
3) Now, again 2nd stack se data 1st stack me daalda.

```
Void InsertAtBottom(stack<int> &st, int x) {
    stack<int> temp;
    while(not st.empty()) {
        int curr = st.top();
        st.pop();
        temp.push(curr);
    }
    st.push(x)
    while(not temp.empty()) {
        int curr = temp.top();
        temp.pop();
        st.push(curr);
    }
}
int main() {
    InsertAtBottom(st,100)
```

Time complexity
Space complexity
O(n)

FREEMIND

Date

Page

⇒ To Solve recursively :-

⇒ Base case me new element push karda
Baaki code same.

```
Void f(st6, int x) {
    if (st.empty()) {
        st.push(x);
        return;
    }
    int curr = st.top();
    st.pop();
    f(st, x);
    st.push(curr);
}
int main() {
    f(st, 100);
    ...
}
```

Q3) Insert at any Idx.

⇒ Remove (size - idx) elements from stack 1 to 2.
⇒ push new element in stack 1
⇒ push back elements again from stack 2 to 1.

Sol)

```
Void InsertAt(stack<int> &st, int x, int idx) {
    stack<int> temp;
    int n = st.size();
    int count = 0
    while (count < n-idx) {
        count++;
        int curr = st.top();
        st.pop();
        temp.push(curr);
```

Date

Page

```
    st.push(x);
    while (not temp.empty()) {
        int curr = temp.top();
        temp.pop();
        st.push(curr);
    }
}
int main() {
    InsertAt(st, 100, 1)
    same
}
```

Q) Remove from Bottom { while (st.size() != 1)

```
Void removeAtBottom(stack<int> &st) {
    stack<int> temp;
    while (st.size() != 1) {
        int curr = st.top()
        st.pop();
        temp.push(curr);
    }
    st.pop();
    while (not temp.empty()) {
        int curr = temp.top();
        temp.pop();
        st.push(curr);
    }
}
Void f(stack<int> &st) {
    if (st.size() == 1) {
        st.pop();
        return;
    }
    same
```

// Recursive

```
int curr = st.top();
st.pop();
cout << curr << "\n";
}
```

FREEMIND

Date

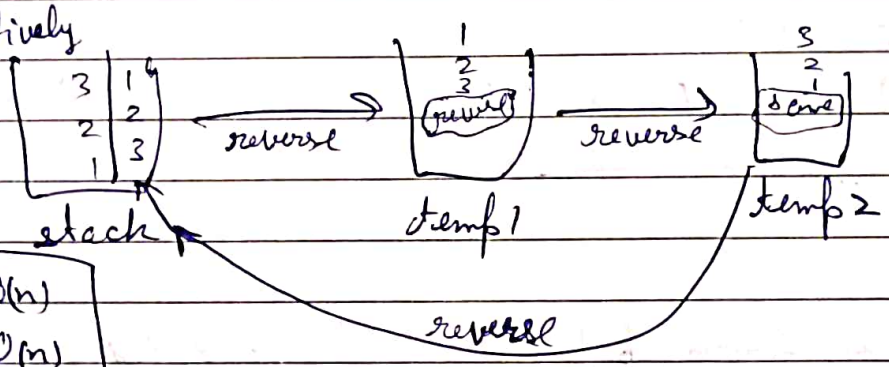
Page

Q) Remove from any index. [n = st.size()]

1) Move (n - idx - 1) elements from stack 1 to 2.
2) Delete the element by st.pop()
3) Move elements back from stack 2 to 1.

Q) Reverse Stack

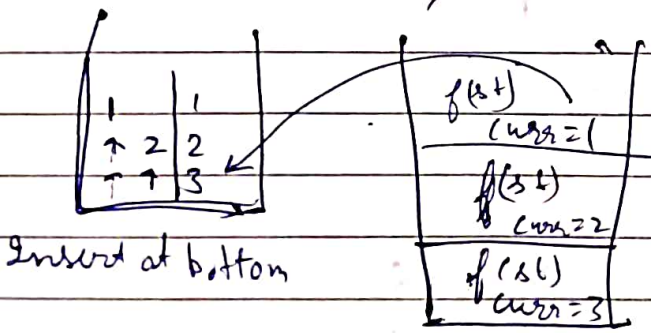
Method 1 ⇒ Iteratively



Time Complexity = O(n)
Space " = O(n)

Method 2 ⇒ Recursively

1) Recursion → call stack (store in temp)
2) Insert at bottom (store back in st)




```
⇒ Void f(stack<int> &st) {
    if (st.empty()) return;
    int curr = st.top();
    st.pop();
    f(st);                    // call stack
    insertatbottom(st, curr);
}
```

Lecture -55

(stacks - Important problems)

(A7) Balanced Bracket Sequence:-

Qs Check whether a given bracket sequence is balanced or not.

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

bool isValid(string str) {
    stack<int> st;
    for (int i = 0; i < str.size(); i++) {
        char ch = str[i];
        if (ch == '(' || ch == '{' || ch == '[') {
            st.push(ch);
        } else {

            if (ch == ')' && !st.empty() && st.top() == '(') {
                st.pop();
            } else if (ch == '}' and !st.empty() and st.top == '{') {
                st.pop();
            } else if (ch == ']' and !st.empty() and st.top == ']') {
                st.pop();
            } else {
                return false;
            }

        }
    } return st.empty
}
int main() {
    string str = "(([[]]))"
    cout << isValid(str) << "\n";
    return;
}
```

FREEMIND

Date

Page

☆ Next Greater Element

(4, 1, 9, 6, 8) Array.

(9, 9, -1, 8, -1) Next greater element

Brute Force Method

```
Void NGE(arr)
int output = {-1, -1, -1, ---- -1};
    for (i=0 ; i < arr.size() ; i++) {
        for (j=i+1; j < arr.size() ; j++) {

            if (arr[i] < arr[j]) {

                output[i] = arr[j];
                break;
            }
        }
```

Stack me idx push karvayenge

Method 2)

```
#include <stacks>

                                                    For pge
1   void
    Vector<int> nge (Vector<int> &arr) {
2
3       int n = arr.size(); //reverse(arr.begin, arr.end())
4       Vector<int> output (n, -1);
5       stack<int> st;
6       for (int i=  st.push(0)
7       for (int i=0 ; i<n ; i++) {
8           while (!st.empty() && arr[i] > arr[st.top()]) {
9               output[st.top()] = arr[i];
10              st.pop();
        }
```

output[st.top()] = n-i-1;

for span of stocks

```
    st.push [i]
}
while (not st.empty()) {
    output [st.top()] = -1;
    st.pop();
} // reverse (arr.begin(), arr.end()); reverse (output.begin(), output.end())
return output;
}
int main() {
    int n;
    cin>> n; Vector<int> v;
    while (n--) {
        int x, i;
        cin>> x;
        v.push_back(x);
    }
    Vector<int> res = nge(v)
    for (int i=0; i<res.size(); i++) {
        cout<< res[i].2<<" ";
    }
    return 0;
}
```

Q) For next Smaller ⇒ 8th Line ⇒ `arr[i] < arr[st.top()]`

Q) For Prev. larger/smaller ⇒

→ Pehle Puri array ko reverse karo and then next greater/smaller nikalo
reverse (arr.begin(), arr.end())

→ And then usse again reverse karde.

Q1 Given a series of N daily price quotes for a stock, we need to calculate the span of the stock's price for all N days. The span of the stock's price in one day is the maximum no. of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

Soln O(N)
O(N)

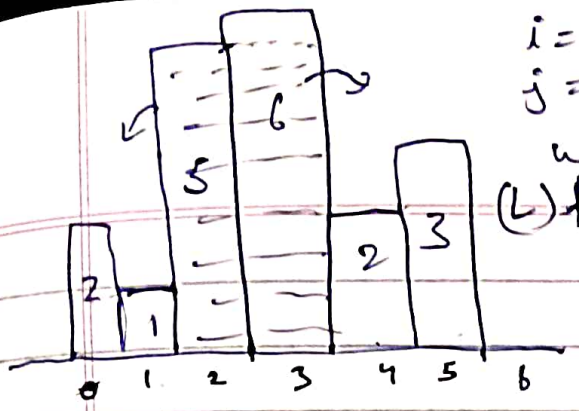
for every index pge & prev. greater index.

[1 1 1 2 1 4 6] → output

#Code

```
Vector<int> pge(Vector<int> &arr) {
    //same as nge {Reverse(arr.begin(), arr.end()) include}
    // output[[st.top()]] = n-i-1; //after reverse indexes will change
}int main() {
    n → Enter
    Vector V → Enter
    for(i=0 to i < res.size(), i++) {
        Cout i < (i - res[i]) << " ";
    }
    return 0;
}
```

Q2 Given an array of integer heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.



i = idx of nse (next smaller element)
j = idx of pse
width ⇒ i - j - 1
(L) height ⇒ bar height

area = height of bar × (i - j - 1)

nse ⇒ 1 6 4 4 6 6
pse ⇒ -1 -1 1 2 1 4

But We have to read the problem two times
So, try in Single read

Method 2

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

int histogram(vector<int>& arr) {
    int n = arr.size();
    stack<int> st;
    int ans = INT_MIN;
    st.push(0);
    for (int i=1; i<n; i++) {
        while (!st.empty() and arr[i] < arr[st.top()]) {
            int el = arr[st.top()];
            st.pop();
            int nsi = i;
            int psi = (st.empty()) ? -1 : st.top();
            ans = max(ans, el * (nsi - psi - 1));
        }
        st.push(i);
    }
    while (not st.empty()) {
        int el = arr[st.top()]
        st.pop()
        int nsi = n
        int psi = (st.empty()) ? (-1) : st.top();
        ans = max(ans, el * (nsi - psi - 1));
    }
    return ans
```

```
int main() {
    int n;
    cin >> n;
    while (n--) {
        int x;
        cin >> x;
        V.push_back(x);
    }
    int ans = histogram(V);
    cout << ans << "\n";
    return 0;
}
```

Lecture -56

(Interview Questions)

Date

Page

Q:- Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

- MinStack() initializes the stack object.
- void push(int val) pushes the element val onto the stack.
- void pop() removes the element on the top of the stack.
- int top() gets the top element of the stack.
- int getMin() retrieves the minimum element in the stack.

You must implement a ~~function~~ solution with O(1) time complexity for each ~~solution~~ function.

Assumption:- Methods pop, top and getMin operations will always be called on non-empty stacks.

Ans →

-5	-4
-1	-6
12	7
6	1
5	-5
10	10

Original (stack)

x - min.

(-5) - (-4)

min. = 10 5 -1 -5

x > min ⇒ x - min > 0

x < min ⇒ x - min < 0

Time, Space O(1)

x - old min. = y

old min = x - y

Original value of Positive Value = Min. + top

~ Enter 1 by 1 ~

leetcode.com

#code

```
#define ll long long int
class Minstack {
public:
    stack<ll> st;
    ll mn;
    Minstack() {
        this->mn = INT_MAX;
    }

    Void push(int Val) {
        if (this->st.empty()) {
            this->mn = Val;
            this->st.push(Val);
        } else {
            this->st.push(val - this->mn)
            if (this->mn > val) {
                this->mn = Val;
            }
        }
    }
    Void pop() {
        if (not this->st.empty()) {
            if (this->st.top() >= 0) {
                this->st.pop();
            } else {
                this->mn = this->mn - this->st.top();
                this->st.pop();
            }
        }
    }
```

```
int top () {
    if (this -> st.size() == 1) {
        return this -> st.top();
    } else if ( this -> st.top() < 0) {
        return this -> mn;
    } else {
        return this -> mn + this -> st.top();
    }
}
int getmin () {
    return this -> min;
}
}
```

Q) Find the minimum no. of brackets that we need to remove to make the given bracket sequence balanced.

Sol)

```
count = 0
stack <char> st;

for (i = 0; i < n; i++) {
    if (str[i] == '(') st.push ('(');
    else {
        if (! st.empty) st.pop();
        else
            count ++;
    }
}
return count;
```

FREEMIND

Date

Page

Infix expressions

BODMAS

(x+z) - y

Postfix expressions

Reverse Polish Notation

A B C D + - + *

A B E - + *

Prefix Expressions

+ - A B C

Evaluation of Postfix Expressions:

2 3 1 * + 9 -

2 (3*1) + 9 -

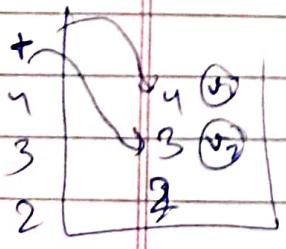
2 3 + 9 -

(2+3) 9 -

5 9 -

5 - 9

-4 Ans



Code

```
#include <stack>
#include <math.h>
using namespace std;
int calc(int v1, int v2, char op){
    if (op == '^'){
        return pow(v1, v2);
    } if (op == '*'){
        return v1 * v2;
    } if (op == '/'){
        return v1 / v2;
    } if (op == '+'){
        return v1 + v2;
    }
    return v1 - v2;
}
```

```
int eval (string &str) {
    stack<int> st;
    for (int i = 0; i < str.size(); i++) {   // prefix Change
        char ch = str[i];
        if (isdigit(ch)) {
            st.push(ch - '0');
        } else {
            int v2 = st.top();
            st.pop();
            int v1 = st.top();
            st.pop();
            st.push(calc(v1, v2, ch));
        }
    }
    return st.top();
}
int main() {
    string str = "231*+9-";   // -9+*132
    cout << eval(str);

    return 0;
}
```

Time, Space O(n)

For prefix

Evaluation of prefix Expressions.

Same as above → for(int i = str.size()-1; i >= 0; i--)

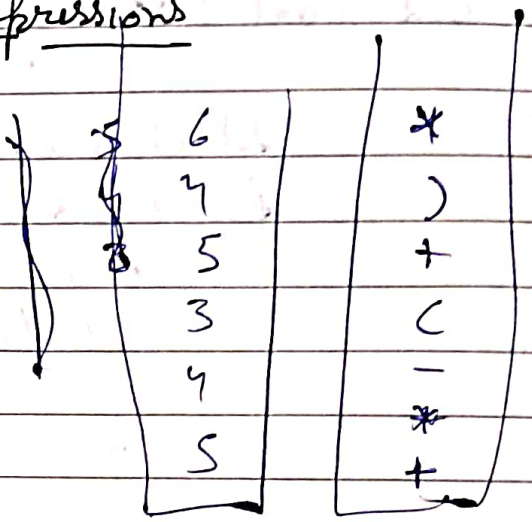
FREEMIND

Date

Page

(A) Evaluation of Infix expressions

5 + 4 * 3 - (5+4) * 6



⇒ no.s or operators dono ko alag-2 stacks me daalenge and jb closing bracket ayegaa operators ko pop krke work karvana start karenge and jb opening bracket aayega tb pop karvana band kardenge.

#Code

```
#include <stack>
#include <math.h>
using namespace std;

int calc(int v1, int v2, char ch) {
    // from prev. ques.
}
int precedence(char ch) {
    if (char == '^') return 3;
    else if (ch == '*' or ch == '/') return 2;
    else if (ch == '+' or ch == '-') return 1;
    else return -1;
}
int eval(string &str) {
    stack<int> nums;
    stack<char> ops;
    for(int i=0 ; i<str.size(); i++) {
        if (isdigit(str[i])) {
            nums.push(str[i] - '0');
        } else
```

```
else if (str[i] == '(') {
    ops.push('(');
} else if (str[i] == ')') {
    while (not ops.empty() and ops.top() != '(') {
        char op = ops.top();
        ops.pop();

        int v2 = nums.top();
        nums.pop();
        int v1 = nums.top();
        nums.pop();
        nums.push(calc(v1, v2, op));
    }
    if (not ops.empty()) ops.pop();
}
else {
    while (not ops.empty() and precedence(ops.top()) >
                                precedence(str[i])) {
        char op = ops.top()
        ops.pop()

        int v2 = nums.top();
        nums.pop();
        int v1 = nums.top();
        nums.pop();
        nums.push(calc(v1, v2, op));
    }
    ops.push(str[i]);
}
while (not ops.empty()) {
    // calculate
}
return nums.top();
}
```

Calculate

FREEMIND

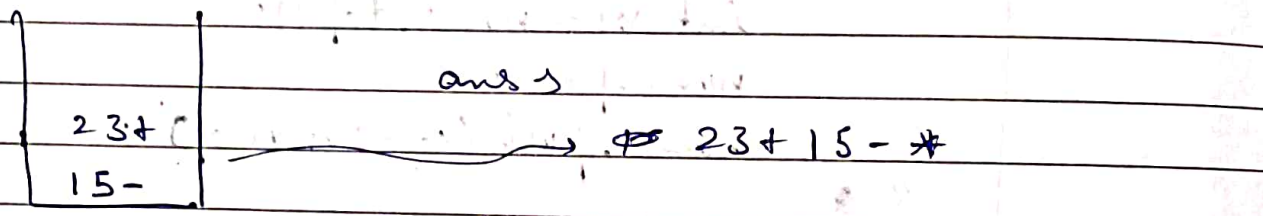
Date

Page

```
int main () {
    string str = "1 + (2*(3-1)) + 2"; // Space hai dena.
    cout << eval (str);
    return 0;
}
```

(*) Conversion of a prefix expression to a postfix expression.

+32-15 —reverse→ 51-23+



#code

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

string eval (string &pre) {
    stack <string> st;
    reverse (pre.begin(), pre.end());
    for (int i=0; i < pre.size(); i++) {
        if (isdigit(pre[i])) {
            st.push (to_string(pre[i] - '0'));
        } else {
            string v1 = st.top();
            st.pop();
            string v2 = st.top();
            st.pop();
            string newexp = v1 + v2 + pre[i];
            st.push (newexp);
        }
    }
    return st.top();
}
```

Time
space O(n)

FREEMIND

Date

Page

```
int main() {
    string s = "*+31-15";
    cout << eval(s);
    return 0;
}
```