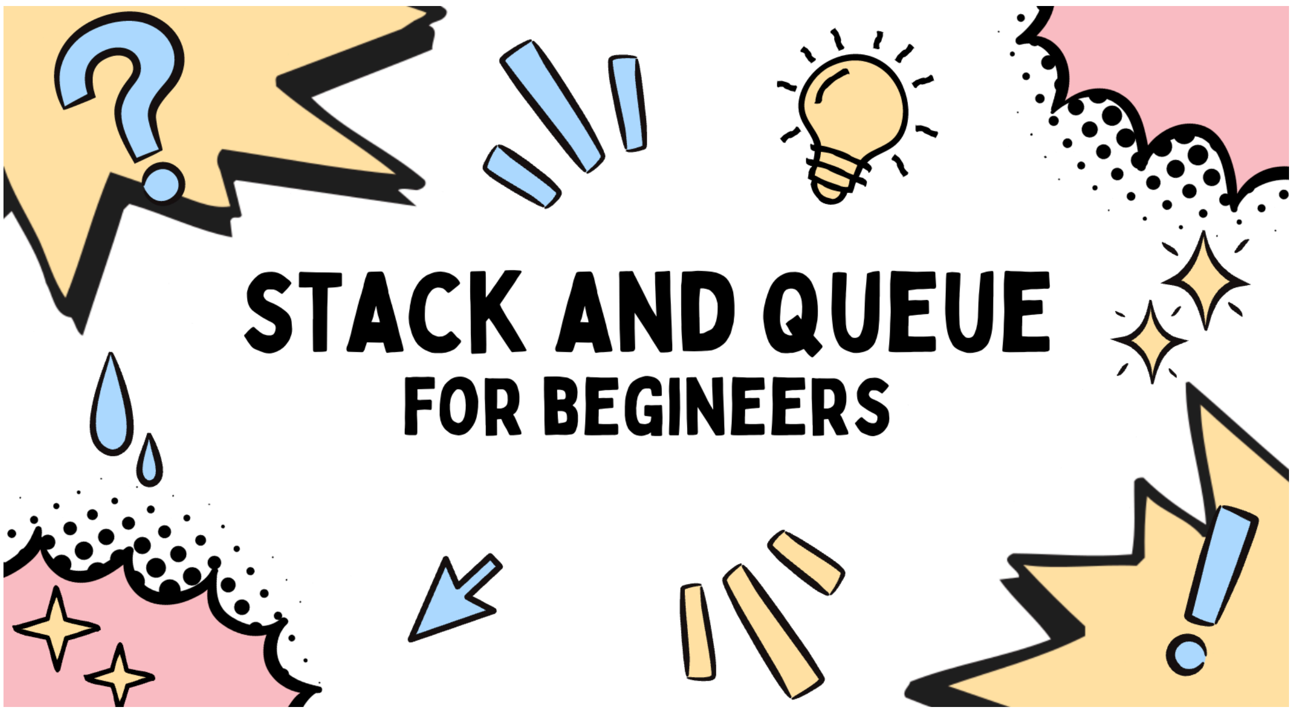
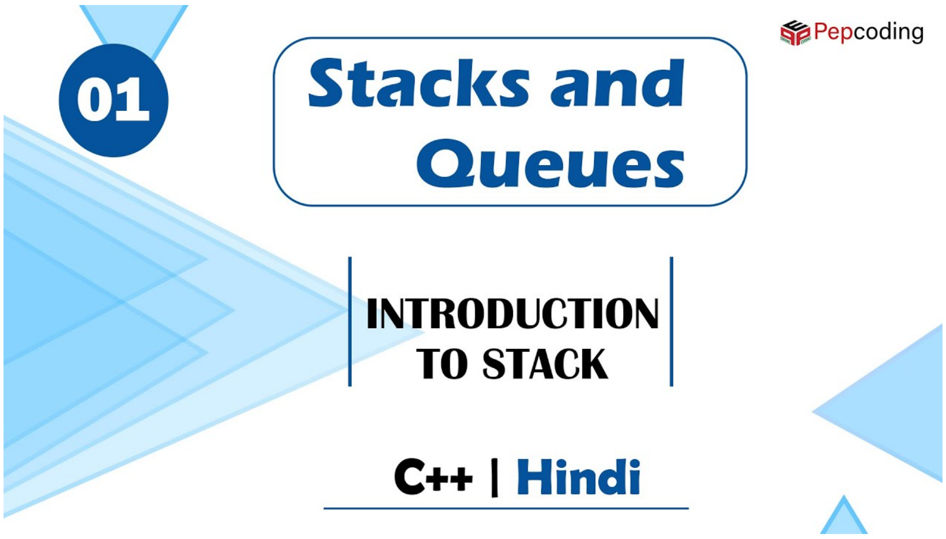


Introduction To Stack And Usage

LEVEL- 1
INTRODUCTION TO STACKS AND ITS USAGE
Pepcoding
For better experience visit
www.nados.pepcoding.com



Duplicate Brackets

Easy

1. You are given a string exp representing an expression. 2. Assume that the expression is balanced i.e. the opening and closing brackets match with each other. 3. But, some of the pair of brackets maybe extra/needless. 4. You are required to print true if you detect extra brackets and false otherwise. e.g.' ((a + b) + (c + d)) -> false (a + b) + ((c + d)) -> true

Constraints

0 <= str.length <= 100

Format

Input

A string str

Output

true or false

Example

Sample Input

```
(a + b) + ((c + d))
```

Sample Output

```
true
```

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;
int main(){
    string  s;
    getline(cin, s);
    stack<char> st;

    for (int i{};i< s.length();i++){
        if(s[i] == ')'){
            if(st.top() == '('){
                cout<<"true"<<endl;
                return 0;
            }
            while(st.top() != '('){
                st.pop();
            }
            st.pop();
        }else{
            st.push(s[i]);
        }
    }

    cout<<"false"<<endl;








   //BEKAR SOLUTION
    // int c{};//no of char till now
    // int cb{};//now char till this bracket

    // for(int i{};i<s.length();i++){
    //     //space is  character
    //     if(st.top() == ')'){
    //         cb = c;
    //     }else if(st.top() == '('){
    //         c = c - cb;
    //         if(c == 0 ){
    //             cout<<"true"<<endl;
    //             return 0;
    //         }
    //         c = cb;
    //         cb = 0;
    //     }else{
    //         c++;
    //     }
    //     st.pop();
    // }
    // cout<<"false"<<endl;

    return 0;
}
```

Balanced Brackets
Easy

1. You are given a string exp representing an expression.
2. You are required to check if the expression is balanced i.e. closing brackets and opening brackets match up well.

e.g.
[(a + b) + {(c + d) * (e / f)}] -> true
[(a + b) + {(c + d) * (e / f)]} -> false
[(a + b) + {(c + d) * (e / f)} -> false
([(a + b) + {(c + d) * (e / f)}] -> false

Constraints

```
0 <= str.length <= 100
```

Format

Input

A string str

Output

true or false

Example

Sample Input

```
[(a + b) + {(c + d) * (e / f)}]
```

Sample Output

```
true
```

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

int main(){
    string s;
    getline(cin,s);
    stack <char> stk;



    for(int i{}; i < s.length() ;i++){
        if(s[i] == '}'){
            if(stk.empty() == true){
                cout<<"false"<<endl;
                return 0;
            }
            if(stk.top() != '{'){
                cout<<"false"<<endl;
                return 0;
            }
            stk.pop();
        }else if(s[i] == ']'){
            if(stk.empty() == true){
                cout<<"false"<<endl;
```

```cpp
            return 0;
        }
        if(stk.top() != '['){
            cout<<"false"<<endl;
            return 0;
        }
        stk.pop();
    }else if(s[i] == ')'){
        if(stk.empty() == true){
            cout<<"false"<<endl;
            return 0;
        }
        if(stk.top() != '('){
            cout<<"false"<<endl;
            return 0;
        }
        stk.pop();
    }else if(s[i] == '[' ||s[i] == '{' ||s[i] == '(' ){
            stk.push(s[i]);
    }


}
if( stk.empty() == false ){
    cout<< "false"<<endl;
}else{
    cout<<"true"<<endl;
}





// int o{};//number of opening brackets
// int c{};//          closing
// for(int i{}; i < s.length() ;i++){
//     if(s[i] == '}'){
//         if(stk.empty() == true){
//             cout<<"false"<<endl;
//             return 0;
//         }
//         c++;
//         while(stk.top() != '{'){
//             if(stk.top() == '(' || stk.top() == '['){
//                 cout<<"false"<<endl;
//                 return 0;
//             }
//             stk.pop();

//         }
//         stk.pop();
//     }else if(s[i] == ']'){
    //          if(stk.empty() == true){
    //              cout<<"false"<<endl;
    //              return 0;
    //          }
    //          c++;
    //          while(stk.top() != '['){
    //              if(stk.top() == '(' || stk.top() == '{'){
    //                  cout<<"false"<<endl;
    //                  return 0;
    //              }
    //              stk.pop();

    //          }
    //          stk.pop();
    //      }else if(s[i] == ')'){
    //          if(stk.empty() == true){
    //              cout<<"false"<<endl;
    //              return 0;
    //          }
    //          c++;
    //          while(stk.top() != '('){
    //              if(stk.top() == '[' || stk.top() == '{'){
    //                  cout<<"false"<<endl;
    //                  return 0;
    //              }
    //              stk.pop();

    //          }
    //          stk.pop();
    //      }else{
    //          if(s[i] == '[' ||s[i] == '{' ||s[i] == '(' ){
    //              o++;
    //          }
    //          stk.push(s[i]);
    //      }
    // }
    // if( o != c){
    //     cout<< "false"<<endl;
    // }else{
    //     cout<<"true"<<endl;
    // }
}
```

Next Greater Element To The Right

Medium

1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing elements of array a.
3. You are required to "next greater element on the right" for all elements of array
4. Input and output is handled for you.

"Next greater element on the right" of an element x is defined as the first element to right of x having value greater than x.
Note -> If an element does not have any element on it's right side greater than it, consider -1 as it's "next greater element on right"
e.g.
for the array [2 5 9 3 1 12 6 8 7]
Next greater for 2 is 5
Next greater for 5 is 9
Next greater for 9 is 12
Next greater for 3 is 12
Next greater for 1 is 12
Next greater for 12 is -1
Next greater for 6 is 8
Next greater for 8 is -1
Next greater for 7 is -1

Constraints

```
0 <= n < 10^5
-10^9 <= a[i] <= 10^9
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
5
3
8
-2
7
```

Sample Output

```
8
8
-1
7
-1
```

```
#include <iostream>
#include<stack>
#include<vector>
using namespace std;
void display(vector<int>a){
    for(int i=0;i<a.size();i++)
    {
        cout<<a[i]<<endl;
    }
}
vector<int> solve(vector<int>arr)
{
  //write your code here



  //space complexity is O(n) because in total while loop will only
run
  //for n times maximum
  //because one element is only being pushed once so it will only
pop for once
  //space complexity is also O(n);




    int n  = arr.size();

    //SIR SOLUTION
    // vector <int> ans(n);
    // stack<int> st;

    // st.push(arr[n-1]);
    // ans[n-1] = -1;
    // for(int i{n-2} ;i >= 0; i--){
    //     while(st.empty() == false && st.top() <= arr[i]){
    //         st.pop();
    //     }

    //     if(st.empty()){
    //         ans[i] = -1;
    //     }else{
    //         ans[i] = st.top();
    //     }
    //     st.push(arr[i]);
    // }
//sir alternative solution

    vector <int> ans(n ,-1);
    stack<int> st;
    st.push(0);
    for(int i{1} ;i < n;i++) {
        while(st.empty() == false && arr[i] > arr[st.top()]){
            ans[st.top()] = arr[i];
            st.pop();
        }
        st.push(i);
    }


   //MY SOLUTION
    // vector<int> ans (n,-1);
    // stack<int> st;
    // st.push(0);
    // for(int i {1}; i < n ; i++){
    //     if(arr[i] <= arr[st.top()] ){
    //         st.push(i);
    //     }else{
    //         while(st.empty() == false && arr[i] >
arr[st.top()]){
    //             ans[st.top()] = arr[i];
    //             st.pop();
    //         }
    //         st.push(i);
    //     }
    // }




//NEXT GREATER TO THE LEFT
    // vector <int> ans(n ,-1);
    // stack<int> st;
    // st.push(arr[0]);

    // for(int i{1} ;i < n;i++) {
    //     while(st.empty() == false && arr[i] >= st.top()){
    //         st.pop();
    //     }
    //     if(st.empty()){

    //     }
    //     else if(arr[i] < st.top()){
    //         ans[i] = st.top();
    //     }
    //     st.push(arr[i]);
    // }
    return ans;


}
```

```
int main(int argc, char **argv)
{
    int n;
    cin>>n;
    vector<int>arr(n,0);
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    vector<int>nge(n,0);
    nge=solve(arr);
    display(nge);
   return 0;
}
```

Stock Span

Easy

1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing the prices of a share on n days.
3. You are required to find the stock span for n days.
4. Stock span is defined as the number of days passed between the current day and the first day before today when price was higher than today.

e.g.
for the array [2 5 9 3 1 12 6 8 7]
span for 2 is 1
span for 5 is 2
span for 9 is 3
span for 3 is 1
span for 1 is 1
span for 12 is 6
span for 6 is 1
span for 8 is 2
span for 7 is 1

Constraints

```
0 <= n < 10^5
-10^9 <= a[i] <= 10^9
```

```
#include <bits/stdc++.h>
using namespace std;
void display(vector<int>a){
    for(int i=0;i<a.size();i++)
    {
        cout<<a[i]<<endl;
    }
}
```

```
vector<int> solve(vector<int>arr)
{
  //write your code here
    int n = arr.size();

    vector <int> ans(n) ;
    stack <int> st ;
    // st.push(n-1);

    // for(int i{n-2} ; i >= 0 ;i--) {
    //     while (st.empty() == false && arr[i] > arr[st.top()]) {
    //         ans[st.top()] = st.top()-i;
    //         st.pop();
    //     }
    //     st.push(i);
    // }
    // while(st.empty() == false){
    //     ans[st.top()] = st.top()+1;
    //     st.pop();
    // }

//SIR SOLUTION POP ADD PUSH

    st.push(0);
    ans[0] = 1;
    for(int i{1}; i < n ; i++){
        while(st.empty() == false && arr[st.top()] <= arr[i]) {
            st.pop();
        }
        if(st.empty()){
            ans[i] = i+1;
        }else{
            ans[i] = i - st.top();
        }
        st.push(i);
    }
    return ans;

}
int main(int argc, char **argv){
    int n;
    cin>>n;
    vector<int>arr(n,0);
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    vector<int>span(n,0);
    span=solve(arr);
    display(span);
   return 0;
}
```

Largest Area Histogram

Hard

1. You are given a number n, representing the size of array a. 2. You are given n numbers, representing the height of bars in a bar chart. 3. You are required to find and print the area of largest rectangle in the histogram. e.g. for the array [6 2 5 4 5 1 6] -> 12

Constraints

0 <= n < 20 0 <= a[i] <= 10

Format

Input

Input is managed for you

Output

A number representing area of largest rectangle in histogram

Example

Sample Input

```
7
6
2
5
4
5
1
6
```

Sample Output

```
12
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(int argc, char **argv)
{
    int n;
    cin>>n;
    vector<int>arr(n,0);
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    //write your code here
    //see sir solution in the video little different then mine

//MY SOLUTION
    //from left to right
    vector <int> ans(n,1 );
    stack<int> st;
    st.push(0);

    for(int i{1}; i < n ;i++) {
        while (st.empty() == false && arr[st.top()] >= arr[i] ) {
            ans[i] += ans[st.top()];
            st.pop();
        }
        st.push(i);
    }

    while(st.empty() == false){
        st.pop();
    }

    //from right to left
    vector <int> ans1(n,1);
    st.push(n-1);
    for(int i{n-2}; i >=0  ;i--) {
        while (st.empty() == false && arr[st.top()] >= arr[i] ) {
            ans1[i] += ans1[st.top()];
            st.pop();
        }
        st.push(i);
    }
    vector <int> finalans(n,0 );
    for(int i{} ;i<n;i++){
        finalans[i] = ans[i]+ans1[i] -1;
        finalans[i] = finalans[i] * arr[i]; //-1 because it was
counting itself twice
    }
    int max = finalans[0];
    for(int i{1} ;i < n ;i++){
        if(finalans[i] > max){
            max = finalans[i];
        }
    }

    cout<<max<<endl;
   return 0;
}
```

Sliding Window Maximum

Hard

1. You are given a number n, representing the size of array a.
2. You are given n numbers, representing the elements of array a.
3. You are given a number k, representing the size of window.
4. You are required to find and print the maximum element in every window of size k.

e.g.
for the array [2 9 3 8 1 7 12 6 14 4 32 0 7 19 8 12 6] and k = 4, the answer is [9 9 8 12 12 14 14 32 32 32 32 19 19 19]

Constraints

```
0 <= n < 100000
-10^9 <= a[i] <= 10^9
0 < k < n
```

Format

Input

Input is managed for you

Output

Maximum of each window in separate line

Example

Sample Input

```
81350
-838923590
2088124597
-100312021
897598134
1120321254
715463751
1960723337
-1292124189
1792804848
967503840
-1916103136
312580715
1132719396
1195946209
62312811
163167577
1625575782
```

……..

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

int main(){
    int n;
    cin >> n;
    vector<int> arr(n);
    for(int i = 0; i < n; i++){
        cin >> arr[i];
    }
    int k;
    cin >> k;

    // write your code here
    vector<int> nge(n, arr.size());
    stack<int> st;

    st.push(n-1);
    for(int i{n-2} ;i >= 0 ; i--){
        while(st.empty() == false && arr[st.top()] <= arr[i]){
            st.pop();
        }
        if(st.empty()){

        }else{
            nge[i] = st.top();
        }
        st.push(i);
    }

    int j{};
    for(int i {} ;i <= n-k ;i++){
        if(j<i){
            j=i;
        }
        while(nge[j] < i+k){
            j = nge[j];
        }

        cout<<arr[j]<<endl;
    }











    cout<<endl;

    return 0;
}
```

Infix Evaluation

Easy

1. You are given an infix expression.
2. You are required to evaluate and print it's value.

Constraints

```
1. Expression is balanced
2. The only operators used are +, -, *, /
3. Opening and closing brackets - () - are used to impact
precedence of operations
4. + and - have equal precedence which is less than * and
/. * and / also have equal precedence.
5. In two operators of equal precedence give preference
to the one on left.
6. All operands are single digit numbers.
```

Format

Input

Input is managed for you

Output

Value of infix expression

Example

Sample Input

```
2 + 6 * 4 / 8 - 3
```

Sample Output

```
2
```

```
#include <bits/stdc++.h>
// #include <ctype.h>
using namespace std;

int prec (char a) {
    if(a == '+' || a == '-') {
        return 1;
    }else if(a == '*' || a == '/'){
        return 2;
    }
}
int operate(int a ,int b, char c){
    if (c == '+'){
        return a + b;
    }else if(c == '-') {
        return a - b;
    }else if(c == '*') {
        return a * b;
    }else if(c == '/') {
        return a / b;
    }
}
```

```cpp
int infix_evaluated(string str) {
    stack<int> st1;
    stack<char> st2;

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            if(st2.empty() ){
                st2.push(str[i]);
            }else if (prec(str[i]) > prec(st2.top())){
                st2.push(str[i]);
            }else{
                while(st2.empty() == false && st2.top() !='(' &&
prec(str[i]) <= prec(st2.top())) {
                    int n1 = st1.top(); st1.pop();
                    int n2 = st1.top(); st1.pop();
                    char op = st2.top(); st2.pop();
                    int num = operate(n2,n1,op);
                    st1.push(num);

                }
                st2.push(str[i]);
            }
        }else if( str[i] == '(') {
            st2.push(str[i]);
        }else if( str[i] == ')'){
            while (st2.top() != '('){
                int n1 = st1.top(); st1.pop();
                int n2 = st1.top(); st1.pop();
                char op = st2.top(); st2.pop();
                int num = operate(n2,n1,op);
                st1.push(num);
            }
            st2.pop();
        }else if (str[i] != ' '){

            int n = str[i] - '0';
            // cout<<n<<endl;
            st1.push(n);
        }
    }

    while (st2.empty() == false) {
        int n1 = st1.top(); st1.pop();
        int n2 = st1.top(); st1.pop();
        char op = st2.top(); st2.pop();
        int num = operate(n2,n1,op);
        st1.push(num);
    }

    return st1.top();
}



int main(){
    string str;
    getline(cin, str);
    // write your code here
    cout << infix_evaluated(str);

    return 0;
}
```

Infix Conversions

Easy

1. You are given an infix expression.
2. You are required to convert it to postfix and print it.
3. You are required to convert it to prefix and print it.

Constraints

```
1. Expression is balanced
2. The only operators used are +, -, *, /
3. Opening and closing brackets - () - are used to impact
precedence of operations
4. + and - have equal precedence which is less than * and
/. * and / also have equal precedence.
5. In two operators of equal precedence give preference
to the one on left.
6. All operands are single digit numbers.
```

Format

Input

Input is managed for you

Output

postfix
prefix

Example

Sample Input

```
a*(b-c+d)/e
```

Sample Output

```
abc-d+*e/
/*a+-bcde
```

```
#include <bits/stdc++.h>
#include <ctype.h>
using namespace std;

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

    return 0;       //sir says in java compiler doesn't know that
we will only pass these for arguments
                    //it think that it should always return an
integer
}

string postfix(string str) {
    stack<char> st;
    string postf;

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            if(st.empty() ){
                st.push(str[i]);
            }else if (prec(str[i]) > prec(st.top())){
                st.push(str[i]);
            }else{
                while(st.empty() == false && st.top() !='(' &&
prec(str[i]) <= prec(st.top())) {
                    postf = postf + st.top();
                    st.pop();
                }
                st.push(str[i]);
            }
        }else if( str[i] == '(') {
            st.push(str[i]);
        }else if( str[i] == ')'){
            while (st.top() != '('){
                postf = postf + st.top();
                st.pop();
            }
            st.pop();
        }else {
            postf = postf + str[i];
        }
    }

    while (st.empty() == false) {
        postf = postf + st.top();
        st.pop();
    }

    return postf;
}
string prefix(string str) {
    stack<char> st;
    stack<string> pref;
    string e{""};

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            if(st.empty() ){
                st.push(str[i]);
            }else if (prec(str[i]) > prec(st.top())){
                st.push(str[i]);
            }else{
                while(st.empty() == false && st.top() !='(' &&
prec(str[i]) <= prec(st.top())) {
                    string s2 = pref.top()+ e;pref.pop();
                    string s1 = pref.top()+ e;pref.pop();
                    string s = st.top()+s1+s2;
                    pref.push(s);
                    st.pop();
                }
                st.push(str[i]);
            }
        }else if( str[i] == '(') {
            st.push(str[i]);
        }else if( str[i] == ')'){
            while (st.top() != '('){
                string s2 = pref.top()+ e;pref.pop();
                string s1 = pref.top()+ e;pref.pop();
                string s = st.top()+s1+s2;
                pref.push(s);
                st.pop();
            }
            st.pop();
        }else {
            pref.push(str[i]+ e);
        }
    }

    while (st.empty() == false) {
        string s2 = pref.top()+ e;pref.pop();
        string s1 = pref.top()+ e;pref.pop();
        string s = st.top()+s1+s2;
        pref.push(s);
        st.pop();
    }

    return pref.top();
}


//MY WAY ANSWER MATCH WITH INTERNET
// string reverse (string str) {
//  // cout<<"hhh"<<endl;
//   string r{""};
//   int n = str.length();
//   for(int i{n-1};i>=0;i--) {
//       if (str[i] == '('){
//           r += ')';
//       }else if (str[i] == ')'){
//           r += '(';
//       }else{
//       r += str[i];
//       }
//   }
//   // cout<<r<<endl;
//   return r;
// }

// string prefix (string str) {
//   string rstr = reverse(str);
//   string postf = postfix(rstr);
//   string pref = reverse(postf);
//   return pref;
// }

int main(){
    string str;
    getline(cin, str);
    // write your code here
    cout << postfix(str)<<endl;
    cout << prefix(str)<<endl;

    return 0;
}
```

Postfix Evaluation And Conversions

Easy

1. You are given a postfix expression.
2. You are required to evaluate it and print it's value.
3. You are required to convert it to infix and print it.
4. You are required to convert it to prefix and print it.

Note -> Use brackets in infix expression for indicating precedence. Check sample input output for more details.

Constraints

```
1. Expression is a valid postfix expression
2. The only operators used are +, -, *, /
3. All operands are single digit numbers.
```

Format

Input

Input is managed for you

Output

value, a number
infix
prefix

Example

Sample Input

```
264*8/+3-
```

Sample Output

```
2
((2+((6*4)/8))-3)
-+2/*6483
```

```
#include <bits/stdc++.h>
using namespace std;


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

    return 0;       //sir says in java compiler doesn't know that
we will only pass these for arguments
                    //it think that it should always return an
integer
}

int operate(int a ,int b, char c){
    if (c == '+'){
        return a + b;
    }else if(c == '-') {
        return a - b;
    }else if(c == '*') {
        return a * b;
    }else if(c == '/') {
        return a / b;
    }

    return 0;
}

void post_eva_and_conversion(string str) {
    stack <string> infix;
    stack <string> prefix;
    stack <int> poste;
    string e {""};

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            int n1 = poste.top(); poste.pop();
            int n2 = poste.top(); poste.pop();
            int num = operate(n2,n1,str[i]);
            poste.push(num);

            //infix
            string si1 = infix.top();infix.pop();
            string si2 = infix.top();infix.pop();
            string si = "("+si2 +str[i]+si1+")";
            infix.push(si);

            //prefix
            string s1 = prefix.top();prefix.pop();
            string s2 = prefix.top();prefix.pop();
            string s =  str[i] + s2 +s1;
            prefix.push(s);

        }else if (str[i] != ' '){

            int n = str[i] - '0';
            poste.push(n);

            string si = str[i] + e;
            infix.push(si);

            string spre = str[i] + e;
            prefix.push(spre);

        }
    }

    cout<<poste.top()<<endl;
    cout<<infix.top()<<endl;
    cout<<prefix.top()<<endl;

}

int main(){
    string exp;
    getline(cin, exp);

    // write yout code here
    post_eva_and_conversion(exp);
    return 0;
}

/*#include <bits/stdc++.h>
```

```
#include <ctype.h>
using namespace std;

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

    return 0;       //sir says in java compiler doesn't know that
we will only pass these for arguments
                    //it think that it should always return an
integer
}
not needed here
int operate(int a ,int b, char c){
    if (c == '+'){
        return a + b;
    }else if(c == '-') {
        return a - b;
    }else if(c == '*') {
        return a * b;
    }else if(c == '/') {
        return a / b;
    }
}
string postfix(string str) {
    stack<char> st;
    string postf;

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            if(st.empty() ){
                st.push(str[i]);
            }else if (prec(str[i]) > prec(st.top())){
                st.push(str[i]);
            }else{
                while(st.empty() == false && st.top() !='(' &&
prec(str[i]) <= prec(st.top())) {
                    postf = postf + st.top();
                    st.pop();
                }
                st.push(str[i]);
            }
        }else if( str[i] == '(') {
            st.push(str[i]);
        }else if( str[i] == ')'){
            while (st.top() != '('){
                postf = postf + st.top();
                st.pop();
            }
            st.pop();
        }else {
            postf = postf + str[i];
        }
    }

    while (st.empty() == false) {
        postf = postf + st.top();
        st.pop();
    }

    return postf;
}
string prefix(string str) {
    stack<char> st;
    stack<string> pref;
    string e{""};

    for (int i {0} ;i < str.length() ;i++) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            if(st.empty() ){
                st.push(str[i]);
            }else if (prec(str[i]) > prec(st.top())){
                st.push(str[i]);
            }else{
                while(st.empty() == false && st.top() !='(' &&
prec(str[i]) <= prec(st.top())) {
                    string s2 = pref.top()+ e;pref.pop();
                    string s1 = pref.top()+ e;pref.pop();
                    string s = st.top()+s1+s2;
                    pref.push(s);
                    st.pop();
                }
                st.push(str[i]);
            }
        }else if( str[i] == '(') {
            st.push(str[i]);
        }else if( str[i] == ')'){
            while (st.top() != '('){
                string s2 = pref.top()+ e;pref.pop();
                string s1 = pref.top()+ e;pref.pop();
                string s = st.top()+s1+s2;
                pref.push(s);
                st.pop();
            }
            st.pop();
        }else {
            pref.push(str[i]+ e);
        }
    }
    while (st.empty() == false) {
        string s2 = pref.top()+ e;pref.pop();
        string s1 = pref.top()+ e;pref.pop();
        string s = st.top()+s1+s2;
        pref.push(s);
        st.pop();
    }

    return pref.top();
}


//MY WAY ANSWER MATCH WITH INTERNET
// string reverse (string str) {
//  // cout<<"hhh"<<endl;
//  string r{""};
//  int n = str.length();
//  for(int i{n-1};i>=0;i--) {
//      if (str[i] == '('){
//          r += ')';
//      }else if (str[i] == ')'){
//          r += '(';
//      }else{
//      r += str[i];
//      }
//  }
//  // cout<<r<<endl;
//  return r;
// }

// string prefix (string str) {
//  string rstr = reverse(str);
//  string postf = postfix(rstr);
//  string pref = reverse(postf);
//  return pref;
// }

int main(){
    string str;
    getline(cin, str);
    // write your code here
    cout << postfix(str)<<endl;
    cout << prefix(str)<<endl;

    return 0;
} */
```

Prefix Evaluation And Conversions

Easy

1. You are given a prefix expression.
2. You are required to evaluate it and print it's value.
3. You are required to convert it to infix and print it.
4. You are required to convert it to postfix and print it.

Note -> Use brackets in infix expression for indicating precedence. Check sample input output for more details.

Constraints

```
1. Expression is a valid prefix expression
2. The only operators used are +, -, *, /
3. All operands are single digit numbers.
```

Format

Input

Input is managed for you

Output

value, a number
infix
prefix

Example

Sample Input

```
-+2/*6483
```

Sample Output

```
2
((2+((6*4)/8))-3)
264*8/+3-
```

```
#include <bits/stdc++.h>
using namespace std;


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

    return 0;       //sir says in java compiler doesn't know that
we will only pass these for arguments
                    //it think that it should always return an
integer
}

int operate(int a ,int b, char c){
    if (c == '+'){
        return a + b;
    }else if(c == '-') {
        return a - b;
    }else if(c == '*') {
        return a * b;
    }else if(c == '/') {
        return a / b;
    }

    return 0;
}

void post_eva_and_conversion(string str) {
    stack <string> infix;
    stack <string> prefix;
    stack <int> poste;

    string e {""};

    for (int i {str.length()-1} ;i >= 0 ;i--) {
        if(str[i] == '+' ||str[i] == '-' ||str[i] == '*' ||str[i]
== '/'){
            int n1 = poste.top(); poste.pop();
            int n2 = poste.top(); poste.pop();
            int num = operate(n1,n2,str[i]);
            poste.push(num);

            //infix
            string si1 = infix.top();infix.pop();
            string si2 = infix.top();infix.pop();
            string si = "("+si1 +str[i]+si2+")";
            infix.push(si);

            //prefix
            string s1 = prefix.top();prefix.pop();
            string s2 = prefix.top();prefix.pop();
            string s =  s1 + s2 + str[i];
            prefix.push(s);

        }else if (str[i] != ' '){

            int n = str[i] - '0';
            poste.push(n);

            string si = str[i] + e;
            infix.push(si);

            string spre = str[i] + e;
            prefix.push(spre);

        }
    }

    cout<<poste.top()<<endl;
    cout<<infix.top()<<endl;
    cout<<prefix.top()<<endl;

}

int main(){
    string exp;
    getline(cin, exp);

    // write yout code here
    post_eva_and_conversion(exp);
    return 0;
}
```

Celebrity Problem

Easy

1. You are given a number n, representing the number of people in a party.
2. You are given n strings of n length containing 0's and 1's
3. If there is a '1' in ith row, jth spot, then person i knows about person j.
4. A celebrity is defined as somebody who knows no other person than himself but everybody else knows him.
5. If there is a celebrity print it's index otherwise print "none".

Note -> There can be only one celebrity. Think why?

Constraints

```
1 <= n <= 10^4
e1, e2, .. n * n elements belongs to the set (0, 1)
```

Format

Input

Input is managed for you

Output

Index of celebrity or none

Example

Sample Input

```
4
0000
1011
1101
1110
```

Sample Output

```
0
```

```cpp
#include<bits/stdc++.h>
using namespace std;

void celebPrblm(vector<string> &v, int n){
    stack <int > st;
    for (int i {0};i<n;i++){
        st.push(i);
    }

    while(st.size() != 1){
        int a = st.top();st.pop();
        int b = st.top();st.pop();
        if(v[a][b] == '1'){
            st.push(b);
        }else{
            st.push(a);
        }
    }
    int pcell = st.top();
    st.pop();

    for(int i{0};i<n;i++) {
        if (i != pcell) {
            if(v[pcell][i] =='1' ||v[i][pcell] =='0') {
                cout<<"none"<<endl;
                return;
            }
        }
    }

    cout<<pcell<<endl;



    //given answer

    // stack<int> st;
    // for(int i = 0; i < n; i++){
    //     st.push(i);
    // }
    // while(st.size() > 1){
    //     int a = st.top();
    //     st.pop();
    //     int b = st.top();
    //     st.pop();

    //     if(v[a][b] == '1'){
    //         st.push(b);
    //     }else{
    //         st.push(a);
    //     }
    // }
    // int possCel = st.top();
    // st.pop();

    // for(int i = 0; i < n; i++){
    //     if(i != possCel){
    //         if(v[i][possCel] == '0' || v[possCel][i] == '1'){
    //             cout << "none" << endl;
    //             return;
    //         }
    //     }
    // }
    // cout << possCel << endl;
}


int main(){
    int n;
    cin >> n;
    vector<string> s;

    for(int i = 0; i < n; i++){
        string str;
        cin >> str;
        s.push_back(str);
    }
    celebPrblm(s, n);
}
```

Merge Overlapping Interval

Medium

1. You are given a number n, representing the number of time-intervals.
2. In the next n lines, you are given a pair of space separated numbers.
3. The pair of numbers represent the start time and end time of a meeting (first number is start time and second number is end time)
4. You are required to merge the meetings and print the merged meetings output in increasing order of start time.

E.g. Let us say there are 6 meetings
1 8
5 12
14 19
22 28
25 27
27 30

Then the output of merged meetings will belongs
1 12
14 19
22 30

Note -> The given input maynot be sorted by start-time.

Constraints

```
1 <= n <= 10^4
0 <= ith start time < 100
ith start time < ith end time <= 100
```

Format

Input

Input is managed for you

Output

Print a merged meeting start time and end time separated by a space in a line .. print all merged meetings one in each line.

Example

Sample Input

```
6
22 28
1 8
25 27
14 19
27 30
5 12
```

Sample Output

```
1 12
14 19
22 30
```

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<stack>

using namespace std;

bool compare (vector <int> v1,vector<int> v2) {
    return v1[0]<v2[0];
}

void print_stack(stack<vector<int>> &a){
    if(a.empty()) {
        return;
    }
    int x = a.top()[0];
    int y = a.top()[1];
    a.pop();
    print_stack(a);
    cout<<x<<" "<<y<<endl;


}
void mergeOverlappingIntervals(vector<vector<int>>& arr){

    // write your code here
    sort(arr.begin(),arr.end(),compare);
    int n = arr.size();

    stack <vector<int>> ans ;
    ans.push(arr[0]);
    for(int i = 1; i < n ; i++ ){
        if (arr[i][0] <= ans.top()[1]) {
            int a = arr[i][1] > ans.top()[1] ? arr[i][1] :
ans.top()[1];
            ans.top()[1] = a;
        }else{
            ans.push(arr[i]);
        }

    }

   print_stack(ans);
}


int main(){
    int n;
    cin >> n ;

    vector<vector<int>> arr( n , vector<int>(2) );



    // input
    for(int i = 0; i < n ; i++ ){
        cin >> arr[i][0];
        cin >> arr[i][1];
    }


    mergeOverlappingIntervals(arr);

}
```

Smallest Number Following Pattern

Easy

1. You are given a pattern of upto 8 length containing characters 'i' and 'd'. 2. 'd' stands for decreasing and 'i' stands for increasing 3. You have to print the smallest number, using the digits 1 to 9 only without repetition, such that the digit decreases following a d and increases follwing an i. e.g. d -> 21 i -> 12 ddd -> 4321 iii -> 1234 dddiddd -> 43218765 iiddd -> 126543

Constraints

0 < str.length <= 8 str contains only 'd' and 'i'

Format

Input

Input is managed for you

Output

Smallest sequence of digits (from 1 to 9) without duplicacy and following the pattern

Example

Sample Input

`ddddiiii`

Sample Output

`543216789`

```
#include<iostream>
#include <string>
#include <stack>
using namespace std;
//sir solution(similar)

void smallest_number_following_pattern(string str){
    if( str.empty() ) {
        return;
    }
    str += "i";

    stack <char> cs;
    stack <int> s;
    int n = str.length();

    int num {1};


//we are breaking given string in to pair of d(s)i
//thinking of i as breaking point
    for(int i {0} ; i < n ; i++) {
       if(str[i] != 'i'){
           s.push(num);
           num++;
       }else{
            s.push(num);
            num++;
            while(s.empty() == false){
            cout<< s.top();
            s.pop();
        }
       }
    }
}





//my solution

// void print_stack(stack<int> s){
//     // prints stack in reverse

//     if( s.empty() ) {
//         return;
//     }

//     int a = s.top();
//     s.pop();
//     print_stack(s);
//     cout<<a;
// }

// void smallest_number_following_pattern(string str){
//     if( str.empty() ) {
//         return;
//     }
//     str += "i";

//     stack <char> cs;
//     stack <int> ns;
//     int n = str.length();

//     int num {0};

//     for(int i {0} ; i < n ; i++) {
//        if(str[i] != 'i'){
//            cs.push(str[i]);
//        }else{
//             num = i+1;
//             ns.push(num);
//             while(cs.empty() == false){
//             num--;
//             ns.push(num);
//             cs.pop();
//         }
//        }


//     }
//      print_stack(ns); //just to print stack in reverse
// }

int main(){
    string str;
    cin >> str;

    // write your code here
    smallest_number_following_pattern(str);
    return 0;

}
```

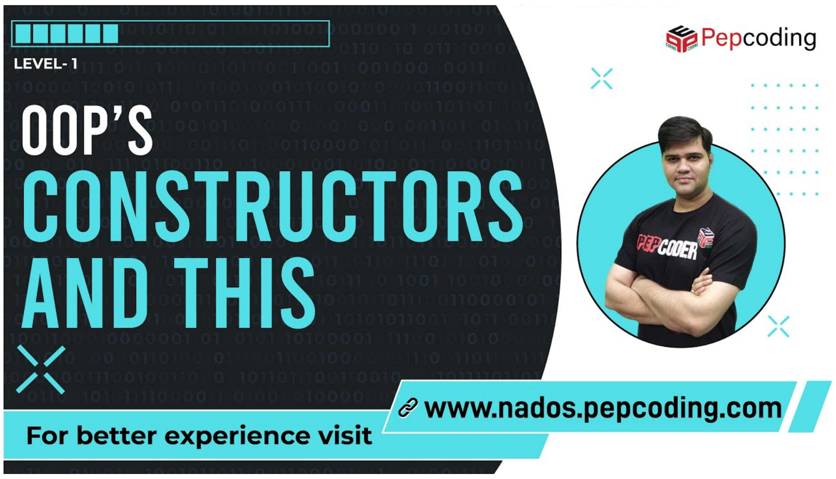
OOP's - Classes And Objects

OOP's - Swap Game 1

OOP's - Swap Game 2

OOP's - Swap Game 3

OOP's - Constructors And This



Normal Stack

Easy

1. You are required to complete the code of our CustomStack class. The class should mimic the behaviour of java.util.Stack class and implement LIFO semantic.
2. Here is the list of functions that you are supposed to complete
   2.1. push -> Should accept new data if there is space available in the underlying array or print "Stack overflow" otherwise.
   2.2. pop -> Should remove and return last data if available or print "Stack underflow" otherwise and return -1.
   2.3. top -> Should return last data if available or print "Stack underflow" otherwise and return -1.
   2.4. size -> Should return the number of elements available in the stack.
   2.5. display -> Should print the elements of stack in LIFO manner (space-separated) ending with a line-break.
3. Input and Output is managed for you.

Constraints

```
None
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
push 10
display
push 20
display
push 30
display
push 40
display
push 50
display
push 60
display
top
pop
display
top
pop
display
top
pop
display
top
pop
display
top
```

```
pop
display
top
pop
quit
```

Sample Output

```
10
20 10
30 20 10
40 30 20 10
50 40 30 20 10
Stack overflow
50 40 30 20 10
50
50
40 30 20 10
40
40
30 20 10
30
30
20 10
20
20
10
10
10

Stack underflow
Stack underflow
```

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

class Stack{

     public:
         int *arr;
         int tos;
         int MaxCapacity;

    public:
        Stack(int size){
            this->MaxCapacity = size;
            this->arr = new int[this->MaxCapacity];
            this->tos = -1;
        }

    public:
        Stack(){
            Stack(10);
        }
```

```
public:
     int size(){
         // write your code here
         return tos+1;
     }

public:
     void push(int data){
         // write your code here
         if(tos == MaxCapacity-1 ) {
           cout<<"Stack overflow" <<endl;
         }else{
           tos++;
           arr[tos] = data;
         }
     }

public:
     int top()
     {
         // write your code here
         if(tos == -1 ) {
           cout<<"Stack underflow" <<endl;
           return -1;
         }else{
           return arr[tos];
         }
     }

public:
     int pop()
     {
         // write your code here
         if(tos == -1 ) {
           cout<<"Stack underflow" <<endl;
           return -1;
         }else{
           int t = tos;
           tos--;
           return arr[t];
         }
     }

 public:
     void display() {
      // write your code here
      int a = tos;
      while(a != -1) {
        cout<<arr[a]<<" ";
        a--;
      }
```

```
            cout<<endl;
        }
    };

    int main(){
        int n;
        cin>>n;
        Stack st(n);
        string str;
        cin>>str;

        while(str != "quit"){
          if(str == "push"){
            int val;
            cin>>val;
            st.push(val);
          } else if(str == "pop"){
            int val = st.pop();
            if(val != -1){
              cout<<val<<endl;
            }
          } else if(str == "top"){
            int val = st.top();
            if(val != -1){
              cout<<val<<endl;
            }

          } else if(str == "size"){
            cout<<st.size()<<endl;
          } else if(str=="display"){
            st.display();
          }
          cin>>str;
        }
    }
```

Dynamic Stack

Easy

1. You are given the code of our CustomStack class which mimics the java.util.Stack class.
2. Here is the list of functions that are written in the class...
   2.1. push -> Accepts new data if there is space available in the underlying array or print "Stack overflow" otherwise
   2.2. pop -> Removes and returns last data if available or print "Stack underflow" otherwise and returns -1
   2.3. top -> Return last data if available or print "Stack underflow" otherwise and returns -1
   2.4. size -> Returns the number of elements available in the stack
   2.5. display -> Print the elements of stack in LIFO manner (space-separated) ending with a line-break.
3. Input and Output are managed for you.
4. You are required to change the body of push function to accept the element even when the data array is completely full. In that case, you are required to reallocate the data array (to twice it's size). You should not print "Stack overflow" ever.

Constraints

```
None
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
push 10
display
push 20
display
push 30
display
push 40
display
push 50
display
push 60
display
top
pop
display
top
pop
display
top
pop
display
top
pop
```

```
display
top
pop
display
top
pop
quit
```

Sample Output

```
10
20 10
30 20 10
40 30 20 10
50 40 30 20 10
60 50 40 30 20 10
60
60
50 40 30 20 10
50
50
40 30 20 10
40
40
30 20 10
30
30
20 10
20
20
10
10
10
```

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

class Stack{

     public:
         int *arr;
         int tos;
         int MaxCapacity;

    public:
         Stack(int size){
             this->MaxCapacity = size;
             this->arr = new int[this->MaxCapacity];
             this->tos = -1;
         }

    public:
         Stack(){
             Stack(10);
         }

    public:
         int size(){
            // write your code here
            return tos+1;
         }

    public:
         void push(int data){

            // write your code here
             if(tos == MaxCapacity-1 ) {
               int *darr = new int[2*(tos+1)];
               for (int i {};i < MaxCapacity ;i++) {
                 darr[i] = arr[i];
               }
               arr = new int[2*(tos+1)];
               MaxCapacity *= 2;
               arr = darr;


             }
               tos++;
               arr[tos] = data;

         }

    public:
         int top()
         {
             // write your code here
             if(tos == -1 ) {
               cout<<"Stack underflow" <<endl;
               return -1;
             }else{
               return arr[tos];
             }
         }

    public:
         int pop()
         {
             // write your code here
             if(tos == -1 ) {
               cout<<"Stack underflow" <<endl;
               return -1;
             }else{
               int t = tos;
               tos--;
               return arr[t];
             }
```

```cpp
        }

    public:
        void display() {
            // write your code here
            int a = tos;
         while(a != -1) {
           cout<<arr[a]<<" ";
           a--;
         }
         cout<<endl;
    }
};

int main(){
    int n;
    cin>>n;
    Stack st(n);
    string str;
    cin>>str;

    while(str!="quit"){
      if(str=="push"){
        int val;
        cin>>val;
        st.push(val);
      } else if(str=="pop"){
        int val = st.pop();
        if(val != -1){
          cout<<val<<endl;
        }
      } else if(str=="top"){
        int val = st.top();
        if(val != -1){
          cout<<val<<endl;
        }

      } else if(str=="size"){
        cout<<st.size()<<endl;
      } else if(str=="display"){
        st.display();
      }
      cin>>str;
    }
}
```

Minimum Stack - I

Easy

1. You are required to complete the code of our MinStack class.
2. As data members you've two stacks available in the class - one for data values and another for minimum values. (This is cryptic - take hint from video)
2. Here is the list of functions that you are supposed to complete
2.1. push -> Should accept new data in LIFO manner
2.2. pop -> Should remove and return data in LIFO manner. If not available, print "Stack underflow" and return -1.
2.3. top -> Should return data in LIFO manner. If not available, print "Stack underflow" and return -1.
2.4. size -> Should return the number of elements available in the stack
2.5. min -> Should return the smallest element available in the stack. If not available, print "Stack underflow" and return -1.
3. Input and Output is managed for you.

Note -> The judge maynot be able to check if all your functions are O(1) in time, but that is what the expectation is.

Constraints

`None`

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
push 10
push 20
push 5
push 8
push 2
push 4
push 11
top
min
pop
top
min
pop
top
min
pop
top
min
pop
top
min
pop
top
min
pop
```

```
top
min
pop
quit
```

Sample Output

```
11
2
11
4
2
4
2
2
2
8
5
8
5
5
5
20
10
20
10
10
10
```

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

class Min_Stack{

    public:
        int *arr;
        int tos;
        int MaxCapacity;
        int mcforms;//mac capacity for min stack
        int *ms;
        int tom;

   public:
        Min_Stack(int size){
            this->MaxCapacity = size;
            this->mcforms = size;
            this->arr = new int[this->MaxCapacity];
            this->tos = -1;
            this->ms = new int[this->MaxCapacity];
            this->tom = -1;
        }

   public:
        Min_Stack(){
            this->MaxCapacity = 10;
            this->mcforms = 10;
            this->arr = new int[this->MaxCapacity];
            this->tos = -1;
            this->ms = new int[this->MaxCapacity];
            this->tom = -1;
        }
    public:
        int size(){
           // write your code here
           return tos+1;
        }
    public:
        void push(int data){

           // write your code here
           //did this with dynamic allocation it would be easy if
you do it
           //by static memory allocation
            if(tos == MaxCapacity-1 ) {
              int *darr = new int[2*(tos+1)];
              for (int i {};i < MaxCapacity ;i++) {
                darr[i] = arr[i];
              }
              arr = new int[2*(tos+1)];
              MaxCapacity *= 2;
              arr = darr;
            }
            if(tom == mcforms-1 ) {
              int *newms = new int[2*(tom+1)];
              for (int i {};i < mcforms ;i++) {
                newms[i] = ms[i];
              }
              ms = new int[2*(tom+1)];
              mcforms *= 2;
              ms = newms;
            }
              tos++;

              if (tos == 0){
                tom = this->tos;
                ms[tom] = tos;
              }else if(data < arr[ms[tom]]) {
                tom++;
                ms[tom] = tos;
              }
              arr[tos] = data;

        }
   public:
        int top()
        {
            // write your code here
            if(tos == -1 ) {
              cout<<"Stack underflow" <<endl;
              return -1;
            }else{
              return arr[tos];

            }
        }

   public:
        int pop()
        {
            // write your code here
            if(tos == -1 ) {
              cout<<"Stack underflow" <<endl;
              return -1;
            }else{
              int t = tos;
              tos--;
              if ( t == ms[tom]) {
                tom--;
              }

              return arr[t];
            }
        }
    public:
        void display() {
            // write your code here
            int a = tos;
         while(a != -1) {
           cout<<arr[a]<<" ";
           a--;
         }
         cout<<endl;
    }
    public:
        int min()
        {
            // write your code here
            if(tos == -1 ) {
              cout<<"Stack underflow" <<endl;
              return -1;
            }else{
              return arr[ms[tom]];
            }
        }
};
```

```
int main(){
  // int n ;
  // cin >>n;
    Min_Stack st;
    string str;
    cin>>str;

    while(str!="quit"){
      if(str=="push"){
        int val;
        cin>>val;
        st.push(val);
      } else if(str=="pop"){
        int val = st.pop();
        if(val != -1){
          cout<<val<<endl;
        }
      } else if(str=="top"){
        int val = st.top();
        if(val != -1){
          cout<<val<<endl;
        }

      } else if(str=="size"){
        cout<<st.size()<<endl;
      } else if(str=="display"){
        st.display();
      }else if (str== "min"){
          cout<<st.min()<<endl;
      }
      cin>>str;
    }
}
```

Minimum Stack - Constant Space

Easy

1. You are required to complete the code of our MinStack class.
2. As data members you've one stack and a min element available in the class. (This is cryptic - take hint from video)
3. Here is the list of functions that you are supposed to complete
   3.1. push -> Should accept new data in LIFO manner.
   3.2. pop -> Should remove and return data in LIFO manner. If not available, print "Stack underflow" and return -1.
   3.3. top -> Should return data in LIFO manner. If not available, print "Stack underflow" and return -1.
   3.4. size -> Should return the number of elements available in the stack.
   3.5. min -> Should return the smallest element available in the stack. If not available, print "Stack underflow" and return -1.

4. Input and Output is managed for you.

Note -> The judge maynot be able to check if all your functions are O(1) in time, but that is what the expectation is. Also, you can only use constant space.r

Constraints

```
None
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
push 10
push 20
push 5
push 8
push 2
push 4
push 11
top
min
pop
top
min
pop
top
min
pop
top
min
pop
top
min
pop
top
min
pop
top
min
pop
quit
```

Sample Output

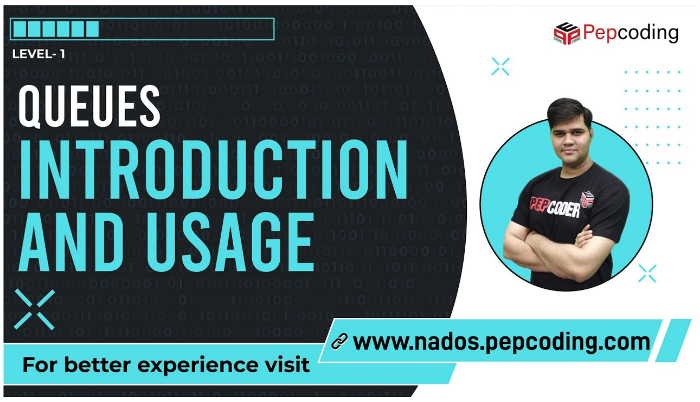
```
11
2
11
4
2
4
2
2
2
8
```

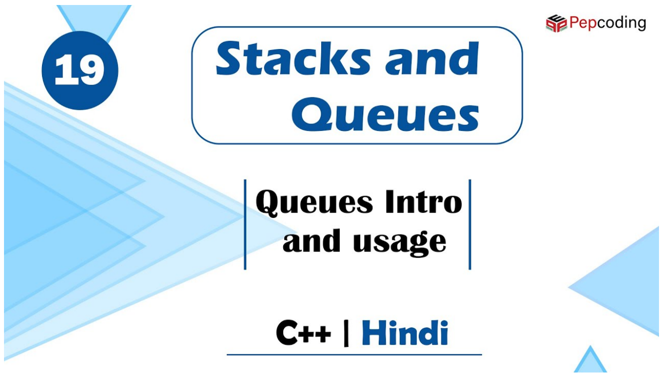
```
5
8
5
5
5
20
10
20
10
10
10
#include<iostream>
#include <string>
#include <stack>

using namespace std;


class MinStack{
    stack<int> data;
    int minVal;


 public:
    int size() {
        // write your code here
        return data.size();
    }

 public:
    void push(int val) {
      // write your code here
      if(data.size() == 0 ){
        minVal = val;
        data.push(val);
      }else if (val < minVal) {
        data.push(val + (val-minVal));//storing a fake smaller
value ->detection
                                       //(can be used to detecte
that it is not normal)
        minVal = val;//original value is stored in min
      }else{
        data.push(val);
      }


    }

 public:
    int pop() {
      // write your code here
      if (data.size() == 0){
        cout<<"Stack underflow"<<endl;
        return -1;
      }else{
        int a = data.top();
        if(data.top() < minVal) {
          a = minVal;
          minVal = 2 * minVal - data.top();
          data.pop();
          return a;
        }else{
          data.pop();
          return a;
        }
      }
    }

 public:
    int top() {
      // write your code here
      if (data.size() == 0){
        cout<<"Stack underflow"<<endl;
        return -1;
      }else{
        if (data.top() < minVal ) {
          return minVal;
        }else{
          return data.top();
        }
      }
    }

 public:
    int min(){
      // write your code here
      if (data.size() == 0){
        cout<<"Stack underflow"<<endl;
        return -1;
      }else{
        return minVal;
      }
    }

};


int main(){
    MinStack st;

    string str;
    cin>>str;

    while(str != "quit"){
```

```cpp
      if(str=="push"){
        int val;
        cin>>val;
        st.push(val);
      }
      else if(str=="pop"){
        int val = st.pop();
        if(val != -1){
          cout<<val<<endl;
        }
      }
      else if(str=="top"){
        int val = st.top();
        if(val != -1){
          cout<<val<<endl;
        }
      }
      else if(str=="size"){
        cout<<st.size()<<endl;
      }
      else if(str== "min" ){
          int val = st.min();
          if(val != -1){
            cout << val << endl;
          }
      }

      cin>>str;

    }

}
//my ans & it is wrong
// #include<iostream>
// #include <string>
// #include <stack>

// using namespace std;


// class MinStack{
//     stack<int> data;
//     int minVal;


//  public:
//     int size() {
//         // write your code here
//         return data.size();
//     }
//  public:
//     void push(int val) {
//       // write your code here
//       if(data.size() == 0 ){
//         minVal = val;
//       }else if (val <= minVal) {
//         data.push(minVal);
//         minVal = val;
//       }
//       data.push(val);

//     }

//  public:
//     int pop() {
//       // write your code here
//       if (data.size() == 0){
//         cout<<"Stack underflow"<<endl;
//         return -1;
//       }else{
//         int a = data.top();
//         if(a == minVal){
//           data.pop();
//           if (data.size() != 0) {
//             minVal = data.top();
//             data.pop();
//           }
//         }else{
//           data.pop();
//         }
//         return a;
//       }
//     }

//  public:
//     int top() {
//       // write your code here
//       if (data.size() == 0){
//         cout<<"Stack underflow"<<endl;
//         return -1;
//       }else{
//         return data.top();
//       }
//     }

//  public:
//     int min(){
//       // write your code here
//       if (data.size() == 0){
//         cout<<"Stack underflow"<<endl;
//         return -1;
//       }else{
```

```
//          return minVal;
//        }
//      }

// };


// int main(){
//     MinStack st;

//     string str;
//     cin>>str;

//     while(str != "quit"){

//       if(str=="push"){
//         int val;
//         cin>>val;
//         st.push(val);
//       }
//       else if(str=="pop"){
//         int val = st.pop();
//         if(val != -1){
//           cout<<val<<endl;
//         }
//       }
//       else if(str=="top"){
//         int val = st.top();
//         if(val != -1){
//           cout<<val<<endl;
//         }
//       }
//       else if(str=="size"){
//         cout<<st.size()<<endl;
//       }
//       else if(str== "min" ){
//           int val = st.min();
//           if(val != -1){
//             cout << val << endl;
//           }
//       }

//       cin>>str;

//     }

// }
```

Queues Intro and usage





Normal Queue

Easy

1. You are required to complete the code of our CustomQueue class. The class should mimic the behaviour of a Queue and implement FIFO semantic.
2. Here is the list of functions that you are supposed to complete
    2.1. add -> Should accept new data if there is space available in the underlying array or print "Queue overflow" otherwise.
    2.2. remove -> Should remove and return value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
    2.3. peek -> Should return value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
    2.4. size -> Should return the number of elements available in the queue.
    2.5. display -> Should print the elements of queue in FIFO manner (space-separated) ending with a line-break.
3. Input and Output is managed for you.

Constraints

```
None
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
add 10
display
add 20
display
add 30
display
add 40
display
add 50
```

```
display
add 60
display
peek
remove
display
peek
remove
display
peek
remove
display
peek
remove
display
peek
remove
display
peek
remove
quit
```

Sample Output

```
10
10 20
10 20 30
10 20 30 40
10 20 30 40 50
Queue overflow
10 20 30 40 50
10
10
20 30 40 50
20
20
30 40 50
30
30
40 50
40
40
50
50
50

Queue underflow
Queue underflow
```

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

class CustomQueue{
    public:
    int *arr;
    int sizeQ;
    int front;
    int arrSize;

    CustomQueue(int cap){
        sizeQ = 0;
        front = 0;
        arr = new int[cap];
        arrSize = cap;
    }

    int size(){
        // write your code here
        return sizeQ;
    }

    void add(int data){
        // write your code here
        if( sizeQ ==0 ){
            front = 0;//front is starting from 0 so this case can
be removed
            arr[front] = data;
            sizeQ++;
        }else if (sizeQ == arrSize) {
            cout<<"Queue overflow"<<endl;
        }
        else {
            arr[( front + sizeQ ) % arrSize] = data;//rear = front
+ sizeQ + 1
            sizeQ++;
        }
    }

    int peek(){
        // write your code here
        if (sizeQ == 0) {
            cout<<"Queue underflow" <<endl;
            return -1;
        }else{
            return arr[front];
        }
    }
```

```
 int remove(){
    // write your code here
    if (sizeQ == 0) {
      cout<<"Queue underflow" <<endl;
      return -1;
    }else if(sizeQ == 1){
      int a = arr[front];
      front = -1;
      sizeQ--;
      return a;
    }else{
      int a = arr[front];
      front = (front + 1) % arrSize ;
      sizeQ--;
      return a;
    }
  }

  void display() {
    // write your code here
    int a = front;
    if (sizeQ == 0) {
      cout <<endl;
    }else{
      for(int i {};i<sizeQ;i++) {
        cout<<arr[a]<<" ";
        a = (a + 1) % arrSize ;
      }
      cout<<endl;
    }
  }
};

int main(){
  int n;
  cin>>n;
  CustomQueue q(n);
  string str;
  cin>>str;

  while(str!="quit"){
    if(str=="add"){
      int val;
      cin>>val;
      q.add(val);
    }
    else if(str=="remove"){
      int val = q.remove();
      if(val != -1){
        cout<<val<<endl;
      }
    }
```

```
        else if(str=="peek"){
            int val = q.peek();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="size"){
            cout<<q.size()<<endl;
        }
        else if(str=="display"){
            q.display();
        }
        cin>>str;
    }
    return 0;
}
```

Dynamic Queue

Easy

1. You are required to complete the code of our CustomQueue class. The class should mimic the behaviour of a Queue and implement FIFO semantic.
2. Here is the list of functions that are written in the class
   2.1. add -> Accepts new data if there is space available in the underlying array or
   print "Queue overflow" otherwise.
   2.2. remove -> Removes and returns value according to FIFO, if available or print
   "Queue underflow" otherwise and return -1.
   2.3. peek -> Returns value according to FIFO, if available or print "Queue
    underflow" otherwise and return -1.
   2.4. size -> Returns the number of elements available in the queue.
   2.5. display -> Prints the elements of queue in FIFO manner (space-separated)
   ending with a line-break.
3. Input and Output is managed for you.

Constraints

None

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
add 10
display
add 20
display
```

```
add 30
display
add 40
display
add 50
display
add 60
display
peek
remove
display
peek
remove
display
peek
remove
display
peek
remove
display
peek
remove
quit
```

Sample Output

```
10
10 20
10 20 30
10 20 30 40
10 20 30 40 50
10 20 30 40 50 60
10
10
20 30 40 50 60
20
20
30 40 50 60
30
30
40 50 60
40
40
50 60
50
50
60
60
60
```

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

class CustomQueue{
    public:
    int *arr;
    int sizeQ;
    int front;
    int arrSize;

    CustomQueue(int cap){
        sizeQ = 0;
        front = 0;
        arr = new int[cap];
        arrSize = cap;
    }

    int size(){
        // write your code here
        return sizeQ;
    }

    void add(int data){
        // write your code here
        if( sizeQ ==0 ){
            front = 0;//front is starting from 0 so this case can
be removed
            arr[front] = data;
            sizeQ++;
        }else if (sizeQ == arrSize) {
            int *newarr = new int [2*arrSize];
            int a = front;
            for(int i {};i<sizeQ;i++) {
              newarr[i] = arr[a];
              a = (a + 1) % arrSize ;
            }
            front = 0;
            newarr[sizeQ] = data;
            arr = newarr;
            sizeQ++;
            arrSize *= 2;
        }
        else {
            arr[( front + sizeQ ) % arrSize] = data;//rear = front
+ sizeQ + 1
            sizeQ++;
        }
    }

    int peek(){
        // write your code here
        if (sizeQ == 0) {
            cout<<"Queue underflow" <<endl;
            return -1;
        }else{
            return arr[front];
        }
    }

    int remove(){
        // write your code here
        if (sizeQ == 0) {
            cout<<"Queue underflow" <<endl;
            return -1;
        }else if(sizeQ == 1){
            int a = arr[front];
            front = -1;
            sizeQ--;
            return a;
        }else{
            int a = arr[front];
            front = (front + 1) % arrSize ;
            sizeQ--;
            return a;
        }
    }

    void display() {
        // write your code here
        int a = front;
        if (sizeQ == 0) {
            cout <<endl;
        }else{
            for(int i {};i<sizeQ;i++) {
              cout<<arr[a]<<" ";
              a = (a + 1) % arrSize ;
            }
            cout<<endl;
        }
    }
};

int main(){
    int n;
    cin>>n;
    CustomQueue q(n);
    string str;
    cin>>str;

    while(str!="quit"){
        if(str=="add"){
            int val;
            cin>>val;
            q.add(val);
        }
        else if(str=="remove"){
            int val = q.remove();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="peek"){
            int val = q.peek();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="size"){
            cout<<q.size()<<endl;
        }
        else if(str=="display"){
            q.display();
        }
        cin>>str;
    }
    return 0;
}
```

Queue To Stack Adapter - Push Efficient

Easy

1. You are required to complete the code of our QueueToStackAdapter class.
2. As data members you've two queues available - mainQ and helperQ. mainQ is to contain data and helperQ is to assist in operations. (This is cryptic - take hint from video)
3. Here is the list of functions that you are supposed to complete
   3.1. push -> Should accept new data in LIFO manner.
   3.2. pop -> Should remove and return data in LIFO manner. If not available, print
    Stack underflow" and return -1.
   3.3. top -> Should return data in LIFO manner. If not available, print "Stack
   underflow" and return -1.
   3.4. size -> Should return the number of elements available in the stack.
4. Input and Output is managed for you.

Note -> push and size should work in constant time. pop and top should work in linear time.

Constraints

```
Note -> push and size should work in constant time. pop
and top should work in linear time.
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
push 10
push 20
push 5
push 8
push 2
push 4
push 11
top
size
pop
top
size
pop
top
size
pop
top
size
pop
top
size
pop
top
size
pop
top
size
pop
quit
```

Sample Output

```
11
7
11
4
6
4
2
5
2
8
4
8
5
3
5
20
2
20
10
1
10
```

```cpp
#include <iostream>
#include <exception>
#include <queue>
using namespace std;

class QueueToStackAdapter{
    public:
    queue<int> mainQ;
    queue<int> helperQ;

    int size(){
        // write your code here
        return mainQ.size();
    }

    void push(int data){
        // write your code here
        mainQ.push(data);
    }

    int top(){
        // write your code here
        if (mainQ.size() == 0) {
            cout<<"Stack underflow"<<endl;
            return -1;
        }else{
            int s = mainQ.size();
            for(int i{};i < s-1 ;i++) {
                helperQ.push(mainQ.front());
                mainQ.pop();
            }
            int a = mainQ.front();
            helperQ.push(mainQ.front());
            mainQ.pop();
            swap(mainQ,helperQ);
            return a;
        }
    }

    int pop(){
        // write your code here
        if (mainQ.size() == 0) {
            cout<<"Stack underflow"<<endl;
            return -1;
        }else{
            int s = mainQ.size();
            for(int i{};i < s-1 ;i++) {
                helperQ.push(mainQ.front());
                mainQ.pop();
            }
            int a = mainQ.front();
            mainQ.pop();
            swap(mainQ,helperQ);
            return a;
        }
    }
};

int main(){
    QueueToStackAdapter st;
    string str;
    cin>>str;

    while(str!="quit"){
        if(str=="push"){
            int val;
            cin>>val;
            st.push(val);
        }
        else if(str=="pop"){
            int val = st.pop();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="top"){
            int val = st.top();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="size"){
            cout<<st.size()<<endl;
        }
        cin>>str;
    }
    return 0;
}
```

Queue To Stack Adapter - Pop Efficient

Easy

1. You are required to complete the code of our QueueToStackAdapter class.
2. As data members you've two queues available - mainQ and helperQ. mainQ is to contain data and helperQ is to assist in operations. (This is cryptic - take hint from video)
3. Here is the list of functions that you are supposed to complete
    3.1. push -> Should accept new data in LIFO manner.
    3.2. pop -> Should remove and return data in LIFO manner. If not available, print "Stack underflow" and return -1.
    3.3. top -> Should return data in LIFO manner. If not available, print "Stack underflow" and return -1.
    3.4. size -> Should return the number of elements available in the stack.
4. Input and Output is managed for you.

Note -> pop, top and size should work in constant time. push should work in linear time.

Constraints

```
Note -> pop, top and size should work in constant time.
push should work in linear time.
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
push 10
push 20
push 5
push 8
push 2
push 4
push 11
top
size
pop
top
size
pop
top
size
pop
top
size
pop
top
size
pop
top
size
pop
quit
```

Sample Output

```
11
7
11
4
6
4
2
5
2
8
4
```

```
8
5
3
5
20
2
20
10
1
10
```

```cpp
#include <iostream>
#include <exception>
#include <queue>
using namespace std;

class QueueToStackAdapter{
    public:
    queue<int> mainQ;
    queue<int> helperQ;

    int size(){
        // write your code here
        return mainQ.size();
    }

    void push(int data){
        // write your code here
        helperQ.push(data);
        while(mainQ.size() != 0) {
            helperQ.push(mainQ.front());
            mainQ.pop();
        }
        swap(mainQ,helperQ);



        ////////OR
        // while(mainQ.size() != 0) {
        //     helperQ.push(mainQ.front());
        //     mainQ.pop();
        // }
        // mainQ.push(data);
        // while(helperQ.size() != 0) {
        //     mainQ.push(helperQ.front());
        //     helperQ.pop();
        // }

    }
```

```
 int top(){
        // write your code here
        if(mainQ.size() == 0){
            cout<<"Stack underflow"<<endl;
            return -1;
        }else{
            return mainQ.front();
        }
    }
    int pop(){
        // write your code here
        if(mainQ.size() == 0){
            cout<<"Stack underflow"<<endl;
            return -1;
        }else{
            int a = mainQ.front();
            mainQ.pop();
            return a;
        }
    }
};

int main(){
    QueueToStackAdapter st;
    string str;
    cin>>str;

    while(str!="quit"){
        if(str=="push"){
            int val;
            cin>>val;
            st.push(val);
        }
        else if(str=="pop"){
            int val = st.pop();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="top"){
            int val = st.top();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="size"){
            cout<<st.size()<<endl;
        }
        cin>>str;
    }
    return 0;
}
```

Stack To Queue Adapter - Add Efficient

Easy

1. You are required to complete the code of our StackToQueueAdapter class. The class should mimic the behaviour of a Queue and implement FIFO semantic.
2. Here is the list of functions that are written in the class
   2.1. add -> Accepts new data if there is space available in the underlying array or print "Queue overflow" otherwise.
   2.2. remove -> Removes and returns value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
   2.3. peek -> Returns value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
   2.4. size -> Returns the number of elements available in the queue.
3. Input and Output is managed for you.

Constraints

```
Note -> add and size should work in constant time. remove and peek should work in linear time.
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
add 10
add 20
add 30
add 40
add 50
remove
remove
add 60
add 70
peek
remove
peek
remove
peek
remove
peek
remove
peek
remove
peek
remove
quit
```

Sample Output

```
10
20
30
```

```
30
40
40
50
50
60
60
70
70
Queue underflow
Queue underflow
```

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

class StackToQueueAdapter{
    public:
    stack<int> mainS;
    stack<int> helperS;

    int size(){
        // write your code here
        return mainS.size();
    }

    void add(int data){
        // write your code here
        mainS.push(data);
    }

    int peek(){
        // write your code here
        if (mainS.size() == 0) {
            cout<<"Queue underflow"<<endl;
            return -1;
        }else{
            while(mainS.size() != 1)  {
                helperS.push(mainS.top());
                mainS.pop();
            }
            int a = mainS.top();
            while(helperS.size() != 0)  {
                mainS.push(helperS.top());
                helperS.pop();
            }
            return a;
        }
    }

    int remove(){
        // write your code here
        if (mainS.size() == 0) {
            cout<<"Queue underflow"<<endl;
            return -1;
        }else{
            while(mainS.size() != 1)  {
                helperS.push(mainS.top());
                mainS.pop();
            }
            int a = mainS.top();
            mainS.pop();
            while(helperS.size() != 0)  {
                mainS.push(helperS.top());
                helperS.pop();
            }
            return a;
        }
    }
};

int main(){
    StackToQueueAdapter q;
    string str;
    cin>>str;

    while(str!="quit"){
        if(str=="add"){
            int val;
            cin>>val;
            q.add(val);
        }
        else if(str=="remove"){
            int val = q.remove();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="peek"){
            int val = q.peek();
            if(val != -1){
                cout<<val<<endl;
            }
        }
        else if(str=="size"){
            cout<<q.size()<<endl;
        }
        cin>>str;
    }
    return 0;
}
```

Stack To Queue Adapter - Remove Efficient

Easy

1. You are required to complete the code of our StackToQueueAdapter class. The class should mimic the behaviour of a Queue and implement FIFO semantic.
2. Here is the list of functions that are written in the class
   2.1. add -> Accepts new data if there is space available in the underlying array or print "Queue overflow" otherwise.
   2.2. remove -> Removes and returns value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
   2.3. peek -> Returns value according to FIFO, if available or print "Queue underflow" otherwise and return -1.
   2.4. size -> Returns the number of elements available in the queue.
3. Input and Output is managed for you.

Constraints

```
Note -> remove, peek and size should work in constant
time. add should work in linear time.
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
add 10
add 20
add 30
add 40
add 50
remove
remove
add 60
add 70
peek
remove
peek
remove
peek
remove
peek
remove
peek
remove
peek
remove
quit
```

Sample Output

```
10
20
30
```

```
30
40
40
50
50
60
60
70
70
Queue underflow
Queue underflow
```

```cpp
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

class StackToQueueAdapter {
public:
  stack <int> mainS;
  stack <int> helperS;

  int size() {
  // write your code here
    return mainS.size();
  }

  void add(int val) {
// write your code here
    while(mainS.size() != 0){
      helperS.push(mainS.top());
      mainS.pop();
    }
    mainS.push(val);
    while(helperS.size() != 0){
      mainS.push(helperS.top());
      helperS.pop();
    }
  }

  int Remove() {
   // write your code here
   if (mainS.size() == 0) {
     cout<< "Queue underflow"<<endl;
     return -1;
   }else{
     int a = mainS.top();
     mainS.pop();
     return a;
   }
  }
int peek() {
    // write your code here
    if (mainS.size() == 0) {
     cout<< "Queue underflow"<<endl;
     return -1;
   }else{
     return mainS.top();
   }
  }
};

int main(){
    string str;
  StackToQueueAdapter qu;
  while (true) {
    getline(cin, str);
    if (str[0] == 'q') {
      break;
    }
    else if (str[0] == 'a') {
      string ss = str.substr(4, 2);
      int n = stoi(ss);
      qu.add(n);
    }
    else if (str[0] == 's') {
      cout << qu.size() << endl;
    }
    else if (str[0] == 'r') {
      int val = qu.Remove();
      if (val != -1) {
        cout << val << endl;
      }
    } else if (str[0] == 'p') {
      int val = qu.peek();
      if (val != -1) {
        cout << val << endl;
      }
    }

  }
  return 0;
}
```

Two Stacks In An Array

Easy

1. You are required to complete the code of our TwoStack class. The class should implement LIFO behaviours for two stacks in the same array.
2. Here is the list of functions that you are supposed to complete
   2.1. push1, push2 -> Should accept new data for appropriate stack if there is

space available in the underlying array or print "Stack overflow" otherwise.
2.2. pop1, po2 -> Should remove and return data from appropriate stack if available or print "Stack underflow" otherwise and return -1.
2.3. top1, top2 -> Should return data from appropriate stack if available or print "Stack underflow" otherwise and return -1.
2.4. size1, size2 -> Should return the number of elements available in appropriate stack.
3. Input and Output is managed for you.

Constraints

```
1. All fns should be constant in time.
2. Memory should be optimally used i.e. one of the stacks
can use more elements in array but overflow should not
happen before whole array is used.
```

Format

Input

Input is managed for you

Output

Output is managed for you

Example

Sample Input

```
5
push1 10
push1 20
push2 30
push2 40
push2 50
push1 60
top1
pop1
top1
pop1
top2
pop2
top2
pop2
top1
pop1
top2
pop2
quit
```

Sample Output

```
Stack overflow
20
20
10
10
50
50
40
40
Stack underflow
Stack underflow
```

```
30
30
#include<iostream>
#include<bits/stdc++.h>
#include<vector>

using namespace std;

   class TwoStack {
    public:
    vector<int> data;
    int tos1;
    int tos2;

    TwoStack(int cap) {
      // write your code here
      data.resize(cap);
      tos1 = -1;
      tos2 = cap;
    }

    int size1() {
      // write your code here
      return tos1+1;
    }

    int size2() {
      // write your code here
      return tos2+1;

    }

    void push1(int val) {
    // write your code here
     if(tos1 == (tos2 - 1)) {
      cout<<"Stack overflow"<<endl;
     }else {
      tos1++;
      data[tos1] = val;
     }
    }

    void push2(int val) {
     // write your code here
     if(tos2 == (tos1 + 1)) {
      cout<<"Stack overflow"<<endl;
     }else {
      tos2--;
      data[tos2] = val;
     }

    }
    int pop1() {
 // write your code here
      if(tos1 == -1) {
        cout<<"Stack underflow"<<endl;
        return -1;
      }else {
        int a = data[tos1];
        tos1--;
        return a ;
      }
    }

    int pop2() {
    // write your code here
      if(tos2 == data.size()) {
        cout<<"Stack underflow"<<endl;
        return -1;
      }else {
        int a = data[tos2];
        tos2++;
        return a ;
      }
    }

    int top1() {
    // write your code here
      if(tos1 == -1) {
        cout<<"Stack underflow"<<endl;
        return -1;
      }else {
        return data[tos1];
      }

    }

    int top2() {
     // write your code here
     if(tos2 == data.size()) {
        cout<<"Stack underflow"<<endl;
        return -1;
      }else {
        return data[tos2];
      }
    }
  };

int main() {
    string str;

    int cap;
    cin>>cap;
    TwoStack st(cap);

  while (true) {
    getline(cin, str);
    if (str[0] == 'q') {
      break;
    }
    else if (str[0] == 'p'&& str[4]=='1') {
      string ss = str.substr(6, 2);
      int n = stoi(ss);
      st.push1(n);

    }
    else if (str[0] == 't'&&str[3]=='1') {
     int tp=st.top1();
      if(tp!=-1) cout<<tp<<endl;
    }
    else if (str[0] == 'p'&&str[3]=='1') {
     int tp=st.pop1();
      if(tp!=-1) cout<<tp<<endl;
    }
    else if (str[0] == 'p'&& str[4]=='2') {
      string ss = str.substr(6, 2);
      int n = stoi(ss);
      st.push2(n);

    }
    else if (str[0] == 't'&&str[3]=='2') {
     int tp=st.top2();
      if(tp!=-1) cout<<tp<<endl;
    }
    else if (str[0] == 'p'&&str[3]=='2') {
      int tp=st.pop2();
      if(tp!=-1) cout<<tp<<endl;
    }

  }
  return 0;

}
```