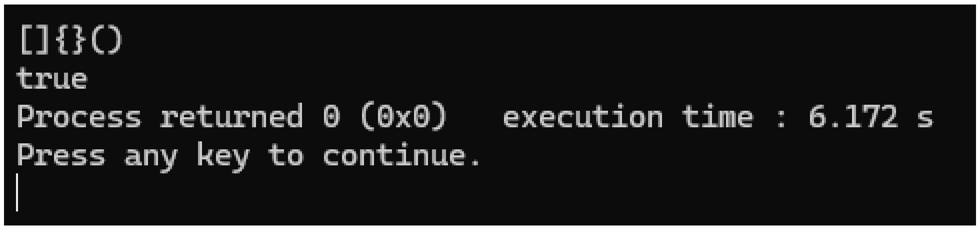
Submitted by IVY SINGH 2401030071
Q1

```
#include <stdio.h>
#include <iostream>
#include <stack>

using namespace std;

bool isValid(string s)
{
    stack<char> st;
    for (char ch : s)
    {
        if (ch == '[' || ch == '{' || ch == '(')
        {
            st.push(ch);
        }
        else
        {
            if (!st.empty() && ((ch == ']' && st.top() == '[') || (ch
== '}' && st.top() == '{') || (ch == ')' && st.top() == '(')))
            {
                st.pop();
            }
            else
            {
                return false;
            }
        }
    }
    if (st.empty())
    {
        return true;
    }
```

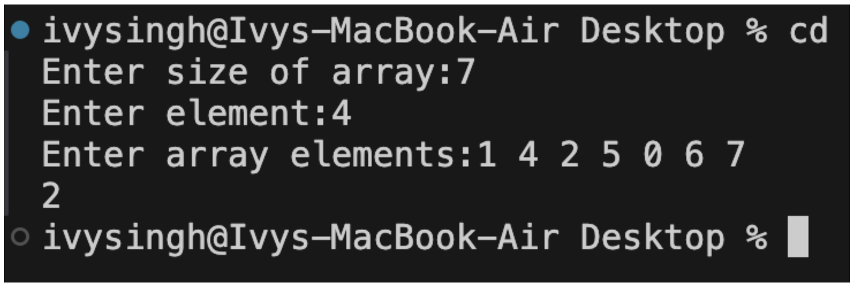
```cpp
    else
    {
        return false;
    }
}
int main()
{
    //string s = "(){}[]";
    string s;
    getline(cin,s);
    cout << boolalpha << isValid(s) << endl;
    return 0;
}
```



```
[]{}()
true
Process returned 0 (0x0)   execution time : 6.172 s
Press any key to continue.
```

Q2

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

int main()
{
    int n,element;
    cout<<"Enter size of array:";
    cin>>n;
    cout << "Enter element:";
    cin >> element;
    int arr[n];
    cout << "Enter array elements:";
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }

    stack<int> st;
    int nextGreater[n];

    for (int i = n - 1; i >= 0; --i)
    {
        while (!st.empty() && arr[st.top()] <= arr[i])
            st.pop();
        nextGreater[i] = st.empty() ? -1 : st.top();
        st.push(i);
    }

    for (int i = 0; i < n; ++i)
    {
        if (arr[i] == element)
        {
            if (nextGreater[i] == -1)
                cout << "Not found\n";
            else
                cout << nextGreater[i] - i << '\n';
            return 0;
        }
    }

    cout << "Element not found\n";
    return 0;
}
```

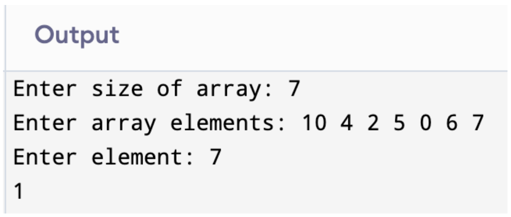


```
ivysingh@Ivys-MacBook-Air Desktop % cd
Enter size of array:7
Enter element:4
Enter array elements:1 4 2 5 0 6 7
2
ivysingh@Ivys-MacBook-Air Desktop % 
```

Q3

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

int main()
{
    int n, element;
    cout << "Enter size of array: ";
    cin >> n;

    int arr[n];
    cout << "Enter array elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << "Enter element: ";
    cin >> element;

    int nextGreater[n];
    for (int i = 0; i < n; i++)
        nextGreater[i] = -1;

    stack<int> st;

    // Traverse from 2n-1 to 0 to simulate circular array
    for (int i = 2 * n - 1; i >= 0; i--) {
        int index = i % n;

        while (!st.empty() && arr[st.top()] <= arr[index])
            st.pop();

        if (i < n && !st.empty())
```

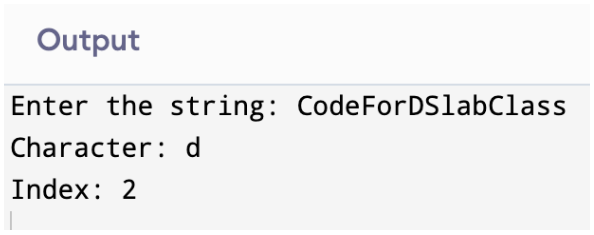
```
            nextGreater[index] = st.top();

        st.push(index);
    }

    // Find index of the input element
    for (int i = 0; i < n; i++) {
        if (arr[i] == element) {
            if (nextGreater[i] == -1) {
                cout << "Not found\n";
            } else {
                int distance = (nextGreater[i] - i + n) % n;
                cout << distance << '\n';
            }
            return 0;
        }
    }

    cout << "Element not found\n";
    return 0;
}
```



```
Enter size of array: 7
Enter array elements: 10 4 2 5 0 6 7
Enter element: 7
1
```

Q4

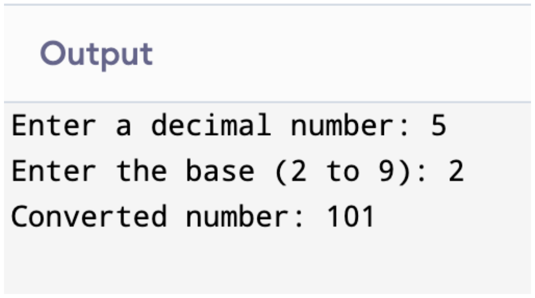
```
#include <iostream>
#include <queue>
using namespace std;

int main() {
    string s;
    cout << "Enter the string: ";
    getline(cin, s);

    int freq[256] = {0};          // Frequency array for ASCII chars
    queue<int> q;                 // Queue stores indices

    for (int i = 0; i < s.length(); i++) {
        char ch = s[i];
        freq[(unsigned char)ch]++;
        q.push(i);

        while (!q.empty() && freq[(unsigned char)s[q.front()]] > 1) {
            q.pop();
        }
    }

    if (q.empty()) {
        cout << "Character: None\n";
        cout << "Index: -1\n";
    } else {
        int idx = q.front();
        cout << "Character: " << s[idx] << '\n';
        cout << "Index: " << idx << '\n';
    }

    return 0;
}
```



```
Enter the string: CodeForDSlabClass
Character: d
Index: 2
```

Q5

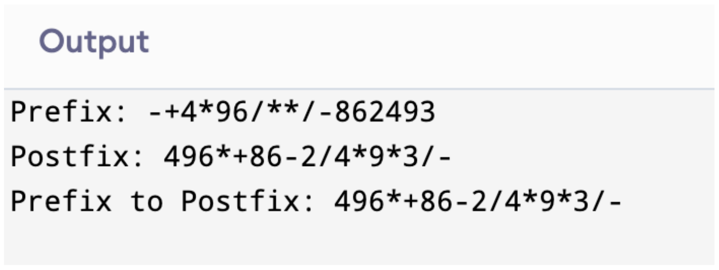
```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    int decimal, base;

    cout << "Enter a decimal number: ";
    cin >> decimal;

    cout << "Enter the base (2 to 9): ";
    cin >> base;

    if (base < 2 || base > 9) {
        cout << "Invalid base! Must be between 2 and 9." << endl;
        return 0;
    }

    if (decimal == 0) {
        cout << "Converted number: 0" << endl;
        return 0;
    }

    stack<int> st;
    int num = decimal;

    while (num > 0) {
        st.push(num % base);
        num /= base;
    }

    cout << "Converted number: ";
    while (!st.empty()) {
        cout << st.top();
        st.pop();
    }
    cout << endl;

    return 0;
}
```



```
Enter a decimal number: 5
Enter the base (2 to 9): 2
Converted number: 101
```

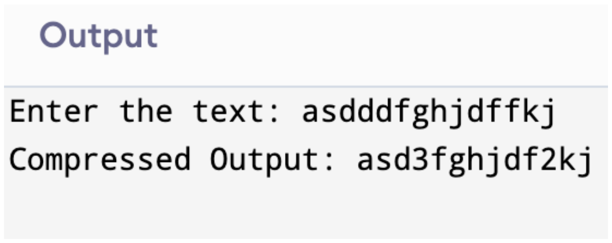
Q6

```
#include <bits/stdc++.h>
using namespace std;
int priority(char op) {
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    return 0;
}
string postfixToPrefix(string s)
{
    stack<string> st;int i=0;
    while(i<s.size())
    {
        if(s[i]>='A' && s[i]<='Z' || s[i]>='a' && s[i]<='z' ||
s[i]>='0' && s[i]<='9')
        st.push(string(1, s[i]));
        else
        {
            string b = st.top(); st.pop();
            string a = st.top(); st.pop();
            string expr = s[i]+ a + b;
            st.push(expr);
        }
        i++;
    }
    return st.top();
}
string prefixToPostfix(string s) {
    stack<string> st;int i=s.size()-1;
    while(i>=0)
    {
        char c = s[i];
        if (s[i]>='A' && s[i]<='Z' || s[i]>='a' && s[i]<='z' ||
s[i]>='0' && s[i]<='9') {
            st.push(string(1, s[i]));
        }
        else
        {
            string a = st.top(); st.pop();
            string b = st.top(); st.pop();
            string expr = a + b + c;
            st.push(expr);
        }
        i--;
    }
    return st.top();
}
string infixTopostfix(string s)
{
    string ans = "";
    stack<char> st;
    int i = 0;
    while (i < s.length())
    {
        char c = s[i];

        if (c == ' ') {
            i++;
            continue;
        }
        if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >=
'0' && c <= '9')) {
            ans += c;
        }
        else if (c == '(') {
            st.push(c);
        }
        else if (c == ')') {
            while (!st.empty() && st.top() != '(') {
                ans += st.top();
                st.pop();
            }
            if (!st.empty()) st.pop();
        }
        else {
            while (!st.empty() && st.top() != '(' && priority(c) <=
priority(st.top())) {
                ans += st.top();
                st.pop();
            }
            st.push(c);
        }

        i++;
    }
    while (!st.empty()) {
        ans += st.top();
        st.pop();
    }

    return ans;
}
int main()
{
    string s="(4 + 9 * 6) - ((8 - 6) / 2 * 4) * 9 / 3";
    string postfix=infixTopostfix(s);
    string prefix=postfixToPrefix(postfix);
    string preTOpro=prefixToPostfix(prefix);
    cout<<"Prefix: "<<prefix<<endl<<"Postfix:
"<<postfix<<endl<<"Prefix to Postfix: "<<preTOpro;
    return 0;
}
```


Output

```
Prefix: -+4*96/**/-862493
Postfix: 496*+86-2/4*9*3/-
Prefix to Postfix: 496*+86-2/4*9*3/-
```

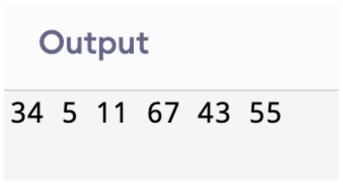
```
Q8
#include <iostream>
#include <queue>
using namespace std;

int main() {
    string input;
    cout << "Enter the text: ";
    getline(cin, input);

    queue<char> q;
    for (char ch : input)
        if (ch != ' ') q.push(ch);  // Enqueue non-space characters
only

    string result;
    while (!q.empty()) {
        char curr = q.front(); q.pop();
        int count = 1;

        while (!q.empty() && q.front() == curr) {
            q.pop();
            count++;
        }

        result += curr;
        if (count > 1)
            result += to_string(count);
    }

    cout << "Compressed Output: " << result << endl;
    return 0;
}
```


Output

```
Enter the text: asdddfghjdffkj
Compressed Output: asd3fghjdf2kj
```

Q9

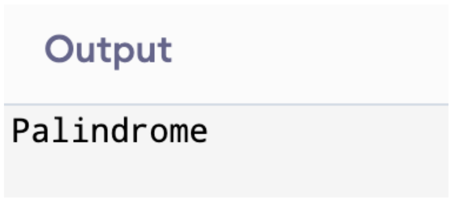
```
#include <iostream>
#include <vector>

using namespace std;

template<typename T>
class Queue {
private:
    vector<T> data;
    size_t frontIndex;

public:
    Queue() : frontIndex(0) {}

    void push(const T& val) {
        data.push_back(val);
    }
    void pop() {
        if (empty()) {
            cout<<"Queue is empty"<<endl;
        }
        frontIndex++;
    }
    T& front() {
        if (empty()) {
            cout<<"Queue id empty"<<endl;
        }
        return data[frontIndex];
    }
    bool empty() const {
        return frontIndex >= data.size();
    }
    size_t size() const {
        return data.size() - frontIndex;
    }
};

int main() {
    Queue<int> Q;
    vector<int> vec={5, 11, 34, 67, 43, 55};
    int n=3;
    for(auto V:vec){
        Q.push(V);
    }
    vector<int> New;
    int nelement=0;
    int count=0;
    while(!Q.empty())
    {
        if(count==n-1){
            nelement=Q.front();
            Q.pop();
            count++;
        }
        else{
            New.push_back(Q.front());
            Q.pop();
            count++;
        }
    }
    New.insert(New.begin(),nelement);
    for(auto v:New){
        cout<<v<<" ";
    }
    return 0;
}
```


Output

```
34 5 11 67 43 55
```

Q10
```
#include <iostream>
#include <vector>

using namespace std;

template<typename T>
class Queue {
private:
    vector<T> data;
    size_t frontIndex;

public:
    Queue() : frontIndex(0) {}

    void push(const T& val) {
        data.push_back(val);
    }
    void pop() {
        if (empty()) {
            cout<<"Queue is empty"<<endl;
        }
        frontIndex++;
    }
    T& front() {
        if (empty()) {
            cout<<"Queue id empty"<<endl;
        }
        return data[frontIndex];
    }
    bool empty() const {
        return frontIndex >= data.size();
    }
    size_t size() const {
        return data.size() - frontIndex;
    }
};
```

```
template<typename T>
class Stack {
public:
    vector<T> ele;
    int index;

    Stack() {
        index = 0;
    }

    void push(T val) {
        ele.push_back(val);
        index++;
    }

    void pop() {
        if (ele.size() == 0) {
            cout << "Stack is Empty" << endl;
            return;
        }
        ele.pop_back();
        index--;
    }

    T top() {
        if (ele.size() == 0) {
            cout << "Stack is Empty" << endl;
            return T();
        }
        return ele[ele.size() - 1];
    }

    int size() {
        return (int)ele.size();
    }

    bool empty() {
        return ele.size() == 0;
    }
};
int main() {
   string s="aabdAdBaA";
   for(int i=0;i<s.length();i++){
    if(s[i]>='A' && s[i]<='Z'){
        s[i]=s[i]+32;
    }
   }
   Stack<char> st;
   Queue<char> ch;
   for(auto c:s){
    st.push(c);
    ch.push(c);
   }
   bool in = true;
   while (!ch.empty())
   {
    if(ch.front()==st.top()){
        ch.pop();
        st.pop();
    }
    else{
        cout<<"NOT a palindrome";
        in = false;
        break;
    }
   }
   if(in==true){
    cout<<"Palindrome";
   }


    return 0;
}
```

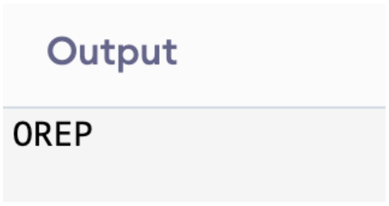


```
Palindrome
```

Q11

```
#include <iostream>
#include <vector>
using namespace std;
template<typename T>
class Stack {
public:
   vector<T> ele;
   int index;

   Stack() {
       index = 0;
   }

   void push(T val) {
       ele.push_back(val);
       index++;
   }

   void pop() {
       if (ele.size() == 0) {
           cout << "Stack is Empty" << endl;
           return;
       }
       ele.pop_back();
       index--;
   }

   T top() {
       if (ele.size() == 0) {
           cout << "Stack is Empty" << endl;
           return T();
       }
       return ele[ele.size() - 1];
   }

   int size() {
       return (int)ele.size();
   }

   bool empty() {
       return ele.size() == 0;
   }
};
int main(){
   string s="ABXNNYPEROYABCDCXT";
   int Xcount=0;
   int Ycount=0;
   Stack<char> st;
   for(auto c:s){
       st.push(c);
   }
```

```
    string New;

    while (!st.empty())
    {
        auto Top=st.top();
        if(Xcount==0 && Top=='X'){
            Xcount++;
        }
        else if(Ycount==0 && Top=='Y' && Xcount==1){
            Ycount++;
        }
        else if(Ycount==1 && Top=='Y'&& Xcount==1){
            Ycount++;
        }
        else if(Xcount==1 && Ycount==1){
            New.push_back(Top);
        }
        else if(Ycount==2 && Top=='X'&& Xcount==1){
            Xcount++;
        }
        else if(Ycount==2 && Xcount==2){
            break;
        }
        st.pop();
    }
    for(auto c:New){
        cout<<c;
    }
}
```


Output

```
OREP
```