Built-in functions are already defined within C++ to do a particular task. Before writing a program you must be aware of important built-in functions as well as libraries so that you will be able to solve complex problems using them.

The abs Function

Absolute value is the value of a number without its sign. The abs function returns the absolute value of a number.

For example, absolute value of 8 is 8 and absolute value of -8 is 8

You might use the abs function to do something like calculate an absolute amount of movement of a character in a game. Take positive values if move it towards right and negative if it move towards left. You can calculate total movements by adding absolute value of movement in each direction.

The function is defined in <cmath> header file in C++ and <math.h> in C.

Question: Find the absolute values of the two numbers entered by the user and display their sum.

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
  int x,y;
  cin>>x>>y;
        int res=abs(x)+abs(y);
 cout<<res;
  //write your code here

}
```

The atoi() function(Converts String to Integer**)**

The atoi() function in C++ interprets the contents of a string and returns its corresponding integer value.

It is defined in <cstdlib> header file.

The atoi() function takes string as parameter, interprets its content as an integral number and returns corresponding value in long int.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    char s[] = "-114";
    double number;
    cout << "Number in String = " << s << endl;
    number = atoi(s);
    cout << "Number in Long Int = " << number;

    return 0;
}
OUTPUT:
Number in String = -114
Number in Long Int = -114
```

Question: Input a number in the form of string from user, add 50 to it and print the value.

```
#include <iostream>
#include <cstdlib>
```

```
using namespace std;
int main()
{
  char s[25];
  cin>>s;
int num=atoi(s);
  cout<<num+50;
  //write your code here

}
```

The exponential(power) function

The **pow()** function computes a base number raised to the power of exponent number.
This function is defined in <cmath> header file
baseexponent = pow(base, exponent)

```
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
double base, exponent, result;

base = 3.4;
exponent = 4.4;
result = pow(base, exponent);

cout << result;

return 0;
}
```

OUTPUT-: 218.025

Question: Write a program to calculate a^b. Take input for a and b from user.

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
  int bas,exponent;
  cin>>bas>>exponent;
  int res=pow(bas,exponent);
  cout<<res;
  //write your code here

}
```

The strlen() function

The strlen() function returns the length of the given string.

The strlen() takes a null terminated byte string str as its argument and returns its length. The length does not include the null character. If there is no null character in the string, the behavior of the function is undefined.
It is defined in <cstring> header file in C++ and <string.h> in C.
Question: Write a program to display the length of the string "Hi! how are you.".

```
#include <iostream>
#include<cstring>
using namespace std;
int main()
{
  char str[]="Hi! how are you.";
cout<<strlen(str);
  //write your code here

}
```

The fmax() and fmin() Function

The fmax() function in C++ takes two arguments and returns the largest among them. If one of the argument is NaN, the other argument is returned.
The fmin() function in C++ takes two arguments and returns the smallest among them. If one of the argument is NaN, the other argument is returned.
Letters are ranked alphabetically and lowercase letters come after uppercase letters. You can also call the max function directly and enter the items that you want to compare into the parentheses as parameters

```
#include <cmath>
using namespace std;
int main()
{
   double x = 56.13, result;
   int y = 89;
   result = fmin(x, y);
   cout << "fmin(x, y) = " << result << endl;
   return 0;
}
```

OUTPUT: fmin(x, y) = 56.13

Question: Find the smallest and largest of the two inputs provided by the user.
If user enters the numbers 2 and 3, then the output should look like:-
Min: 2
Max: 3

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int a,b;
  cin>>a>>b;
  cout<<"Min: "<<fmin(a,b)<<endl;
  cout<<"Max: "<<fmax(a,b);
```

```
  //write your code here

}
```

The strcmp() function

The strcmp() function in C++ compares two null terminated strings. The comparison is done lexicographically.

The strcmp() function takes two arguments: LHS and RHS. It compares the contents of LHS and RHS lexicographically. The sign of the result is the sign of the difference between the first pairs of characters that differ in LHS and RHS.

The behavior of **strcmp()** is undefined if either of LHS or RHS do not point to null terminated strings.

It is defined in <cstring> header file in C++ and <string.h> in C.

The strcmp() function returns:

a positive value if the first differing character in LHS is greater than the corresponding character in RHS.

negative value if the first differing character in LHS is less than the corresponding character in RHS.

0 if LHS and RHS are equal.

Question: Write a program to print the lexicographically (according to the dictionary) greater string. Take two strings as input.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
  char a[50];
  char b[50];
  cin>>a;
  cin>>b;
  int com=strcmp(a,b);
  if(com>0){
    cout<<a;
  }else if(com<0){
    cout<<b;
  }else{
    cout<<"equal";
  }
  //write your code here

}
```

The sort() Function

Sorts the elements in the range (first,last) into ascending order.

sort(a,a+10); will sort array a with 10 elements.

It is included in <algorithm> library

Question: Write a C++ code to sort the given array arr = {40,36,55,75,12,48,63,25} using sort() and then display the elements of the sorted array in separate lines.

```
#include <iostream>
```

```
#include<algorithm>
using namespace std;
int main()
{
  int arr[] = {40,36,55,75,12,48,63,25};
  int s=sizeof(arr)/sizeof(arr[0]);
  //write your code here
  sort(arr,arr+s);
  for(int i=0;i<s;i++){
    cout<<arr[i]<<endl;
  }
}
```


File Handling through C++ Classes

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read from and write to files.

Now the first step is to open the particular file for read or write operation. We can open file by

1. passing file name in constructor at the time of object creation
2. using the open method

#include <fstream> //include for files
ofstream fout; // Creation of ofstream class object
string line;
fout.open("sample.txt"); //open file
fout << line << endl; // Write line in file

Select among the following, the correct C++ code to write a single line to file "sample.txt" and read that line from the file.

A

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
ofstream fout;
string line;
fout.open("sample.txt");
getline(cin, line);
fout << line << endl;
fout.close();
ifstream fin;
fin.open("sample.txt");
while (fin) {
getline(fin, line);
cout << line << endl;
}
fin.close();
return 0;
}
```

B

```
#include <iostream>
```

```
#include <fstream>
using namespace std;
int main()
{
ofstream fout;
string line;
fout.open("sample.txt");
while (fout) {
getline(cin, line);
fout << line << endl;
}
fout.close();
ifstream fin;
fin.open("sample.txt");
while (fin) {
getline(fin, line);
cout << line << endl;
}
return 0;
}
```

C

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
ofstream fout;
string line;
fout.open("sample.txt");
while (fout) {
getline(cin, line);
fout << line << endl;
}
fout.close();
ifstream fin;
while (fin) {
getline(fin, line);
cout << line << endl;
}
fin.close();
return 0;
}
```

D

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
ofstream fout;
string line;
fout.open("sample.txt");
fout.close();
ifstream fin;
fin.open("sample.txt");
while (fin) {
getline(fin, line);
cout << line << endl;
}
fin.close();
return 0;}
```

We have an array of elements
{2, 56, 7, -6, 99, -76, -33, 69, 3, -56, 4, 56, 34, -9, 10, -20, 38, 55}
Print absolute value of each element in a separate line.

```
        #include <iostream>
using namespace std;
int main()
{
  int num[]={2,56,7,-6,99,-76,-33,69,3,-56,4,56,34,-9,10,-20,38,55};
  int s=sizeof(num)/sizeof(num[0]);
  for(int i=0;i<s;i++){
    cout<<abs(num[i])<<endl;
  }
  //write your code here

}
```

Print the value of a after performing following functions:-
a = fmin(fmax(11,22), fmax(abs(-30),20))

```
        #include <iostream>
#include<cmath>
using namespace std;
int main()
{
  int a=fmin(fmax(11,22),fmax(abs(-30),20));
  cout<<a;
  //write your code here

}
```

Write a program that prints all the numbers with absolute value less than 3 in the array arr[] = {-1, 2, -3, 4, 5}.
Each number should be displayed in a separate line.

```
        #include <iostream>
using namespace std;
int main()
{
  int num[]={-1,2,-3,4,5};
  int size=sizeof(num)/sizeof(num[0]);
  for(int i=0;i<size;i++){
    if(abs(num[i])<3){
      cout<<num[i]<<endl;
    }
  }
  //write your code here

}
```

What will be the value of max(min(abs(-63), 25), a[3]) if a = {53, 5, 36, 65, 78, 10}. Output that value.

```
        #include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int a[]={53,5,36,65,78,10};
  int res=max(min(abs(-63), 25), a[3]);
  cout<<res;
  //write your code here

}
```

Create an array of numbers from 125 to 65 (both inclusive) in which each number in the array decreases by five from the previous number. Each number should be displayed in a separate line.

```
        #include <iostream>
using namespace std;
int main() {
    for (int i = 125; i >= 65; i -= 5) {
        cout << i << endl;
    }
    return 0;
}
```

Create an array of numbers from 25 to 75 (both inclusive) where each number in the array increases by two from the previous number. Then print the array with each number printed in a separate line.

```
        #include <iostream>
using namespace std;
int main()
{
     for (int i = 25; i <= 75; i += 2) {
       cout << i << endl;
      }
  // write your code here
}
```

Create an array of numbers from 32 to 89(both inclusive) where each number is increased by three from the previous number. You have to display the following in separate lines:-
1. Length of the array.
2. Largest number in the array.
3. Sum of all the elements of array.

```
#include <iostream>
using namespace std;
int main() {
int sum=0,j=0,i;
```

```
for(i=32;i<=89;i=i+3) {
sum=sum+i; j++; }
cout<<j<<endl;
cout<<i-3<<endl;
cout<<sum<<endl;
return 0; }
```

Vectors in C++

Vectors are sequence containers representing arrays that can change in size.<vector> library needs to be included.

Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements and just as efficiently as in arrays. But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

assign() – It assigns a new value to the vector elements by replacing old ones

push_back() – It pushes the elements into a vector from the back

pop_back() – It is used to pop or remove elements from a vector from the back.

insert() – It inserts new elements before the element at the specified position

erase() – It is used to remove elements from a container from the specified position or range.

swap() – It is used to swap the contents of one vector with another vector of same type and size.

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector<int> g1;
   for (int i = 1; i <= 5; i++)
      g1.push_back(i);
}
```

The above code will create a vector and add elements from 1 to 5 to it.

Question: Create a vector num and add 5 to 10. Then delete the last element and print the vector elements in separate lines.

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
  //write your code here
    vector<int> num;
    for (int i = 5; i <= 10; i++) {
        num.push_back(i);
    }
    if (!num.empty()) {
        num.pop_back();
    }
    for (int n : num) {
        cout << n << endl;
    }
}
```

Maps

Maps are associative containers that store elements formed by a combination of a key value and a mapped value following a specific order.

In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key. The types of key and mapped value may differ and are grouped together in member type value_type, which is a pair type combining both.

Some basic functions associated with Map:

begin() – Returns an iterator to the first element in the map

end() – Returns an iterator to the theoretical element that follows last element in the map

size() – Returns the number of elements in the map

max_size() – Returns the maximum number of elements that the map can hold

empty() – Returns whether the map is empty

pair insert(keyvalue, mapvalue) – Adds a new element to the map

erase(iterator position) – Removes the element at the position pointed by the iterator

erase(const g)– Removes the key value 'g' from the map

clear() – Removes all the elements from the map

insert() - Insert elements with a particular key in the map container

count() - Returns the number of matches to element with key value 'g' in the map.

Create iterator to access map.

map<int, int>::iterator itr; -> This will create an iterator for map. We can access map by key = itr->first and value = itr->second

Question: Create a map using following Key and elements. Then print the map in the format given in the template code.

Now remove all the elements with key less than 4. Again print the map.

Then remove the element with key 7. Print the map again.

```
KEY     ELEMENT
1       400
2       300
3       600
4       200
5       500
6       500
7       100
```

```cpp
#include <iostream>
#include <map>

using namespace std;

void printMap(const map<int, int> &myMap) {
    cout << "KEY\tELEMENT" << endl;
    for (const auto &entry : myMap) {
        cout << entry.first << "\t" << entry.second << endl;
    }
    //cout << endl;
}

int main() {
    map<int, int> myMap = {
```

```
        {1, 400},
        {2, 300},
        {3, 600},
        {4, 200},
        {5, 500},
        {6, 500},
        {7, 100}
    };
    printMap(myMap);
    for (auto itr = myMap.begin(); itr != myMap.end();) {
        if (itr->first < 4) {
            itr = myMap.erase(itr);
        } else {
            ++itr;
        }
    }
    printMap(myMap);
    myMap.erase(7);
    printMap(myMap);
}
```

Lists in C++

Lists are sequence containers that allow non-contiguous memory allocation. As compared to vector, list has slow traversal but once a position has been found insertion and deletion are quick. Normally, when we say a list, we talk about doubly linked list. For implementing a singly linked list, we use forward list.

front() – Returns the value of the first element in the list.
back() – Returns the value of the last element in the list .
push_front(g) – Adds a new element 'g' at the beginning of the list .
push_back(g) – Adds a new element 'g' at the end of the list.
pop_front() – Removes the first element of the list and reduces size of the list by 1.
pop_back() – Removes the last element of the list and reduces size of the list by 1

Question: Write a program that takes two numbers(a and b) as input from the user. Create a list with numbers from a to b(both inclusive). Remove first and last number and print the list with each element in separate line.

```
#include <iostream>
#include <list>
#include<iterator>
using namespace std;
int main() {
    int a, b;
    cin >> a >> b;
    list<int> myList;
    for (int i = a; i <= b; i++) {
        myList.push_back(i);
    }
    if (!myList.empty()) {
        myList.pop_front();
```

```
    }
    if (!myList.empty()) {
        myList.pop_back();
    }
    for (int num : myList) {
        cout << num << endl;
    }
  //write your code here

}
```

Sets in C++

Sets are a type of associative containers in which each element has to be unique because the value of the element identifies it. The elements are stored in ascending order in the set by default. The value of the element cannot be modified once it is added to the set, though it is possible to remove and add the modified value of that element.

Some basic functions associated with Set:

begin() – Returns an iterator to the first element in the set.

end() – Returns an iterator to the theoretical element that follows last element in the set.

size() – Returns the number of elements in the set.

max_size() – Returns the maximum number of elements that the set can hold.

empty() – Returns whether the set is empty.

```
#include <iostream>
#include <set>
#include <iterator>
using namespace std;
int main()
{
  // empty set container
  set <int, greater <int> > set1;

  // insert elements in random order
  set1.insert(40);
  set1.insert(30);
  set1.insert(60);
  set1.insert(50);
  set1.insert(50); // only one 50 will be added to the set
  set1.insert(10);
}
```

Question: Create a set {4,3,6,5,1,2,7,8} in descending order. Remove all the elements greater than 6. Remove element 4. Print the modified set elements in separate lines.

```
#include <iostream>
#include <set>
#include <iterator>

using namespace std;

int main() {
    set<int, greater<int>> set1;
```

```
    set1.insert(4);
    set1.insert(3);
    set1.insert(6);
    set1.insert(5);
    set1.insert(1);
    set1.insert(2);
    set1.insert(7);
    set1.insert(8);
    set<int, greater<int>>::iterator itr;
    for (itr = set1.begin(); itr != set1.end();) {
        if (*itr > 6) {
            itr = set1.erase(itr);
        } else {
            ++itr;
        }
    }
    set1.erase(4);
    for (int num : set1) {
        cout << num << endl;
    }
  //write your code here

}
```

Random Number Generation

It is often useful to generate random numbers to produce simulations or games. One way to generate these numbers in C++ is to use the function rand(). rand() is defined as:
#include <cstdlib>
int rand();

The rand function takes no arguments and returns an integer that is a pseudo-random number between 0 and RAND_MAX. On transformer, RAND_MAX is 2147483647. What is a pseudo-random number? It is a number that is not truly random, but appears random. That is, every number between 0 and RAND_MAX has an equal chance (or probability) of being chosen each time rand() is called. (In reality, this is not the case, but it is close)
Any integer modulus 10 will produce a number from 0-9. In other words, if we divide a number by 10, the remainder has to be from 0 to 9. It's impossible to divide a number by 10 and end up with a remainder bigger than or equal to ten.
We can take rand() % 10 to give us numbers from 0-9. If we want numbers from 1-10 we can now just scale up by adding one. The final result is:
cout << (rand() % 10) + 1 << endl;
Write a code to create an array of numbers from 10 to 20, choose a random number from the array and print.

A

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

```
int main()
{
int arr[11]={10,11,12,13,14,15,16,17,18,19,20};
int number =rand();
cout<<arr[number];
}
```

B

```
#include<cstdlib>
#include<iostream>
using namespace std;
int main()
{
int arr[11]={10,11,12,13,14,15,16,17,18,19,20};
int number =rnd();
cout<<arr[number];
}
```

C

```
#include<cstdlib>
#include<iostream>
using namespace std;
int main()
{
int arr[11]={10,11,12,13,14,15,16,17,18,19,20};
int number =rnd()%11;
cout<<arr[number];
}
```

D

```
#include<cstdlib>
#include<iostream>
using namespace std;
int main()
{
int arr[11]={10,11,12,13,14,15,16,17,18,19,20};
int number =rand()%11;
cout<<arr[number];
}
```

Create a vector with 5 elements {5,10,15,20,25}. Display Size of the vector before and after pushing the elements. Then display the vector with each element in a separate line.

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> vec = {};
```

```
    cout<<vec.size()<< endl;
    vec.push_back(5);
    vec.push_back(10);
    vec.push_back(15);
    vec.push_back(20);
    vec.push_back(25);
  cout<<vec.size()<<endl;
    for (int num : vec) {
        cout << num << endl;
    }
}
```

Write a program that takes 2 3x3 matrices as input from the user. You have to add these 2 matrices and then output the resultant matrix with each element in a separate line.

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<vector<int>> matrix1(3, vector<int>(3));
    vector<vector<int>> matrix2(3, vector<int>(3));
    vector<vector<int>> result(3, vector<int>(3));
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> matrix1[i][j];
        }
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> matrix2[i][j];
        }
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << result[i][j] << endl;
        }
    }
}
```

Create a map using following data:

Name	Age
ram	23
sam	36

mohan 24
karan 34

Take a name as input from user and print his age if that name is present in the map. Print "Not found" if not present.

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main() {
    map<string, int> ages = {
        {"ram", 23},
        {"sam", 36},
        {"mohan", 24},
        {"karan", 34}
    };
    string name;
    cin >> name;
    if (ages.find(name) != ages.end()) {
        cout <<ages[name] << endl;
    } else {
        cout << "Not found" << endl;
    }
}
```

Create a list of numbers input by user. Keep taking input till user inputs 0. Sort the list in descending order and print all the elements of list in separate lines.

```
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;
int main() {
    list<int> numberList;
    int num;
    while (true) {
        cin >> num;
        if (num == 0) {
            break;
        }
        numberList.push_back(num);
    }
    numberList.sort(greater<int>());
    for (int num : numberList) {
        cout << num << endl;
    }
}
```

You are given a two dimensional array of size N X M but it is not completely full. The program takes q queries from the user. Each query has 2 values x and y. It should print the integer stored at the position (x, y). The error message "ERROR!" should be printed if the requested position is empty. The indexing starts from 1.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, i, j;
    vector<vector<int>> vect;
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cin >> m;
        vector<int> vect_m;
        for (j = 0; j < m; j++)
        {
            int value;
            cin >> value;
            vect_m.push_back(value);
        }
        vect.push_back(vect_m);
    }
    int no_of_Queries;
    cin >> no_of_Queries;
    int x, y;
    for (i = 0; i < no_of_Queries; i++)
    {
        cin >> x >> y;
        if (x <= vect.size() && y <= vect[x - 1].size())
        {
            cout << vect[x - 1][y - 1] << endl;
        }
        else
        {
            cout << "ERROR!" << endl;
        }
    }
}
```

You are given a 6 X 6 2D array.
For example: If we create an hourglass using the number 1 within an array full of zeros, it may look like this:
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Actually there are many hourglasses in the array above.

PULKIT

The sum of an hourglass is the sum of all the numbers within it. The sum for the hourglass above is 7.
In this problem you have to print the largest sum among all the hourglasses in the array.

```
#include<bits/stdc++.h>
using namespace std;
void Sum(int arr[6][6]) {
    int maxSum = INT_MIN;

    for (int i = 0; i <= 3; i++) {
        for (int j = 0; j <= 3; j++) {
            int currentSum = 0;
            currentSum += arr[i][j] + arr[i][j + 1] + arr[i][j + 2];
            currentSum += arr[i + 1][j + 1];
            currentSum += arr[i + 2][j] + arr[i + 2][j + 1] + arr[i + 2][j + 2];
            if (currentSum > maxSum) {
                maxSum = currentSum;
            }
        }
    }
    cout<<maxSum<<endl;
}

int main() {
    int arr[6][6];
    for (int arr_i = 0; arr_i < 6; arr_i++) {
        for (int arr_j = 0; arr_j < 6; arr_j++) {
            int temp;
            cin >> temp;
            arr[arr_i][arr_j] = temp;
        }
    }
    Sum(arr);
}
```

Given 2 BitSets B1 and B2 of size N where all bits in both BitSets are initialized to 0, perform a series of M operations. After each operation, print the number of set bits in the respective BitSets as two space-separated integers on a new line. Any element having a bit value of 1 is called a set bit.)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm> // for std::count
using namespace std;

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

    vector<bool> x(n, false);
```

```
    vector<bool> y(n, false);

    vector<vector<bool>> bs(3);
    bs[1] = x;
    bs[2] = y;

    while (m-- > 0) {
        string command;
        int a, b;
        cin >> command >> a >> b;

        if (command == "AND") {
            for (int i = 0; i < n; ++i) {
                bs[a][i] = bs[a][i] && bs[b][i];
            }
        } else if (command == "OR") {
            for (int i = 0; i < n; ++i) {
                bs[a][i] = bs[a][i] || bs[b][i];
            }
        } else if (command == "XOR") {
            for (int i = 0; i < n; ++i) {
                bs[a][i] = bs[a][i] != bs[b][i];
            }
        } else if (command == "FLIP") {
            bs[a][b] = !bs[a][b];
        } else if (command == "SET") {
            bs[a][b] = true;
        }

        int set1 = count(bs[1].begin(), bs[1].end(), true);
        int set2 = count(bs[2].begin(), bs[2].end(), true);

        cout << set1 << " " << set2 << endl;
    }

    return 0;
}
```

A string containing only parentheses is balanced if the following is true:

1. If it is an empty string
2. If A and B are balanced, AB is balanced,where A and B are strings of parenthesis
3. If A is balanced, (A) and {A} and [A] are also balanced, where A is string of parenthesis

Examples of some correctly balanced strings are: "{}()", "[{()}]", "({()})"
Examples of some unbalanced strings are: "{}(", "({)}", "[[", "}{" etc.
Given a string, determine if it is balanced or not. Output "true" or "false"

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

```
using namespace std;

bool areParanthesisBalanced(string expr)
{
    stack<char> s;
    char x;

    for (int i = 0; i < expr.length(); i++) {
        if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{') {
            s.push(expr[i]);
            continue;
        }

        if (s.empty())
            return false;

        switch (expr[i]) {
        case ')':
            x = s.top();
            s.pop();
            if (x == '{' || x == '[')
                return false;
            break;

        case '}':
            x = s.top();
            s.pop();
            if (x == '(' || x == '[')
                return false;
            break;

        case ']':
            x = s.top();
            s.pop();
            if (x == '(' || x == '{')
                return false;
            break;
        }
    }

    return s.empty();
}

int main()
{
    string expr;
    cin >> expr;
    if (areParanthesisBalanced(expr))
```

CLKIT

```
        cout << "true\n";
    else
        cout << "false\n";
    return 0;
}
```

Comparators are used to compare two objects.
In this challenge, you'll create a comparator and use it to sort an array.
The Player class is provided for you in your editor. It has fields: a name (Type String) and a score (Type integer)
Given an array of Player objects, write a comparator that sorts them in order of decreasing score; if 2 or more players have the same score, sort those players alphabetically by name.
Display the sorted player objects as given in the expected output.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Player
{
    char name[20];
    int score;
};
int compare(const void *p, const void *q)
{
    struct Player *player1 = (struct Player *)p;
    struct Player *player2 = (struct Player *)q;

    if (player1->score == player2->score) {
        return strcmp(player1->name, player2->name);
    } else {
        return player2->score - player1->score;
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    struct Player ar[n];
    int i;

    for(i = 0; i < n; i++)
    {
        scanf("%s %d", ar[i].name, &ar[i].score);
    }

    qsort(ar, n, sizeof(ar[0]), compare);
```

```
    for(i = 0; i < n; i++)
    {
        printf("%s %d\n", ar[i].name, ar[i].score);
    }

    return 0;
}
```

You are given N integers. You need to find the maximum number of unique integers among all the possible contiguous subarrays of size M.
Example : Lets say the array is
5 3 5 2 3 2
And M = 3 (contiguous sub array size)
So , S1 = {5,3,5} – has 2 unique numbers
S2 = {3,5,2} – has 3 unique numbers
S3 = {5,2,3} – has 3 unique numbers
S4 = {2,3,2} – has 2 unique numbers
So, The maximum amount of unique numbers among all possible contiguous subarrays is 3.

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

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

    map<int, int> mp;
    vector<int> arr(n);

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

    int uniqueCount = 0, maxUniqueCount = 0;

    for (int i = 0; i < m; i++) {
        if (mp[arr[i]] == 0) {
            uniqueCount++;
        }
        mp[arr[i]]++;
    }

    maxUniqueCount = uniqueCount;

    for (int i = m; i < n; i++) {
        if (mp[arr[i - m]] == 1) {
            uniqueCount--;
        }
        mp[arr[i - m]]--;
```

```
        if (mp[arr[i]] == 0) {
            uniqueCount++;
        }
        mp[arr[i]]++;

        maxUniqueCount = max(maxUniqueCount, uniqueCount);
    }

    cout << maxUniqueCount << endl;

    return 0;
}
```

For an entered array and search key value, write a program to implement a linear searching and print 'Element is present at index <value>', where value=index position of search key in the array, if search key is present otherwise print 'Element is not present in array'.

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

int Linearsearch(int arr[], int n, int search)
{
    for (int i = 0; i < n; i++)
    {
        if (arr[i] == search)
        {
            return i;
        }
    }

    return -1;
}

int main(void)
{
    int arr[10], search, n;
    cin >> n;

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

    cin >> search;
    int result = Linearsearch(arr, n, search);

    (result == -1) ? cout << "Element is not present in array" : cout <<
"Element is present at index " << result;

    return 0;}
```

For an entered array and search key value, write a program to implement a binary searching and print 'Element is present at index <value>', where value=index position of search key in the array, if search key is present otherwise print 'Element is not present in array'.

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

int binarySearch(int arr[], int l, int r, int search)
{
    while (l <= r)
    {
        int mid = l + (r - l) / 2;

        if (arr[mid] == search)
            return mid;

        if (arr[mid] < search)
            l = mid + 1;
        else
            r = mid - 1;
    }

    return -1;
}

int main(void)
{
    int arr[10], search, n;
    cin >> n;

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

    cin >> search;
    int result = binarySearch(arr, 0, n - 1, search);

    (result == -1) ? cout << "Element is not present in array" : cout <<
"Element is present at index " << result;

    return 0;
}
```

Write a program to find an one of the missing integer of the entered array and print the result of it.

Note: An array range is 1 to n+1 integer numbers.There are no duplicates in the array. Missing number is a missing number from element range {1, 2, ...n+1} of array.

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

int findMissingNo(int arr[], int n) {
    int total = (n + 1) * (n + 2) / 2;
    for (int i = 0; i < n; i++) {
        total -= arr[i];
    }
    return total;
}

int main() {
    int arr[20], n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int missno = findMissingNo(arr, n);
    cout << missno;
}
```

Write a program to search an entered key value in a sorted and pivoted array. The output should be displayed the 'Key not found', if an entered key is not found in the array otherwise displayed 'Index: value' ,where value=index position of search key in the array.

```cpp
#include <stdio.h>

int search(int arr[], int l, int h, int key) {
    while (l <= h) {
        int mid = (l + h) / 2;

        if (arr[mid] == key)
            return mid;

        if (arr[l] <= arr[mid]) {
            if (key >= arr[l] && key <= arr[mid])
                h = mid - 1;
            else
                l = mid + 1;
        }
        else {
            if (key >= arr[mid] && key <= arr[h])
                l = mid + 1;
            else
                h = mid - 1;
        }
    }
    return -1;
}
```

```c
int main() {
    int n, arr[20], key;
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    scanf("%d", &key);

    int index = search(arr, 0, n - 1, key);

    if (index != -1)
        printf("Index: %d\n", index);
    else
        printf("Key not found");

    return 0;
}
```

PULKIT

Given a string S, partition S such that every string of the partition is a palindrome. Print all possible palindrome partitioning of S. A single character is also considered palindromic.

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

bool isPalindrome(string& s, int start, int end) {
    while (start < end) {
        if (s[start] != s[end])
            return false;
        start++;
        end--;
    }
    return true;
}

void partitionUtil(string s, int start, vector<string>& path,
vector<vector<string>>& result) {
    if (start == s.length()) {
        result.push_back(path);
        return;
    }

    for (int i = start; i < s.length(); i++) {
        if (isPalindrome(s, start, i)) {
            path.push_back(s.substr(start, i - start + 1));
            partitionUtil(s, i + 1, path, result);
            path.pop_back();
        }
    }
}

void partition(string s, vector<vector<string>>& v) {
    vector<string> path;
    vector<vector<string>> result;

    partitionUtil(s, 0, path, result);

    v = result;
}

int main() {
    string str;
    cin >> str;
    vector<vector<string>> partitions;
    partition(str, partitions);
    for (auto& partition : partitions) {
        for (int i = 0; i < partition.size(); i++) {
            cout << partition[i] << " ";
        }
```

```
        cout << endl;
    }

    return 0;
}
```

A Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once. Given an undirected graph, the task is to check if a Hamiltonian path is present in it or not. If a hamiltonian path is present in the graph, then print 1 otherwise print 0.

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

int A[15][15];
int ch = 0;

void ham(int n, int V[], int v) {
    V[v] = 1;
    bool allVisited = true;
    for (int i = 1; i <= n; i++) {
        if (!V[i]) {
            allVisited = false;
            if (A[v][i] == 1)
                ham(n, V, i);
        }
    }
    if (allVisited) {
        ch = 1;
        return;
    }
    V[v] = 0;
}

int main() {
    ch = 0;
    int v, e;
    cin >> v >> e;
    int a, b;

    for(int i = 0; i < v + 1; i++)
        for(int j = 0; j < v + 1; j++)
            A[i][j] = 0;

    for(int i = 0; i < e; i++) {
        cin >> a >> b;
        A[a][b] = A[b][a] = 1;
    }
```

PULKIT

```
    int V[v + 1];
    for(int i = 0; i < v + 1; i++)
        V[i] = 0;

    for(int i = 1; i < v + 1; i++) {
        ham(v, V, i);
        if(ch == 1)
            break;
    }

    if(ch == 1)
        cout << "1\n";
    else
        cout << "0\n";

    return 0;
}
```

Github Link https://github.com/Pulkitbarala/Myperfectice

PULKIT