Q1) Airport security officials have confiscated several items of the passenger at the security checkpoint. All the items have been dumped into a huge box(array). Each item possessed a certain amount of risk(0,1,2). Here is the risk severity of the item representing an array[] of N number of integer values. The risk here is to sort the item based on their level of risk values range from 0 to 2.

Example 1: Input: 7 ----- Value of N
[1,0,2,0,1,0,2] -> Element of arr[0] to arr[N-1], while input each element is separated by new line
Output: 0 0 0 1 1 2 2 -> Element after sorting based on the risk severity.

Example 2:
Input:
10   ----- Value of N
[2,1,0,2,1,0,0,1,2,0] -> Element of arr[0] to arr[N-1], while input each element is separated by new line
Output:
0 0 0 0 0 1 1 1 2 2 2 -> Element after sorting based on the risk severity.

Constraints
0<N<=100
0<=arr[i]<=2

Code Solution:
```
#include<iostream>
 using namespace std;
```

```
int main()
{
  int n;
  cin >> n;
   int arr[n];
   for(int i = 0; i < n; i++)
    {
       cin >> arr[i];
          }
 for(int i = 0; i < n; i++)
  {
    for(int j = i+1; j < n; j++)
       {
         if(arr[i] > arr[j])
             {
               int temp = arr[i];
               arr[i] = arr[j];
               arr[j] = temp;
                       }
                   }
               }
for(int i = 0; i < n; i++)
         {
          cout << arr[i] << " ";
           }
     }
```

Q2) Given N gold wires, each wire has a length associated with it. At a time, only two adjacent small wres assembled at the end of a large wire and the cost of forming is the sum of their length. Find the minimum cost when all wires are assembled to form a single wire.

For Example: Suppose, Arr[] = {7, 6, 8, 6, 1, 1}
{7, 6, 8, 6, 1, 1} - {7, 6, 8, 6, 2}, cost = 2
{7, 6, 8, 6, 2} - {7, 6, 8, 8}, cost = 8
{7, 6, 8, 8} - {13, 8, 8}, cost =13
{13, 8, 8} - {13, 16}, cost = 16
{13, 16} - {29}, cost = 29
2 + 8 + 13 + 16 + 29 = 68
Hence, the minimum cost to assemble all gold wires is : 68
Constraints:
1 <= N <= 30
1<= Arr[] <= 100
Example 1:

Input:

6 -> Value of N, represents size of Arr
7 -> Value of Arr[0], represents length of 1st wire
6 -> Value of Arr[1], represents length of 2nd wire
8 -> Value of Arr[2], represents length of 3rd wire
6 -> Value of Arr[3], represents length of 4th wire
-1 -> Value of Arr[4], represents length of 5th wire
1 -> Value of Arr[5], represents length of 6th wire

Output:

68

Example 2:

Input:

4 -> Value of N, represents size of Arr

12 -> Value of Arr[0], represents length of 1st wire
2 -> Value of Arr[1], represents length of 2nd wire
2 -> Value of Arr[2], represents length of 3rd wire
5-> Value of Arr[3], represents length of 4th wire

Output:
34

Code Solution:

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

 struct MinHeap
 {
  unsigned size;
  unsigned capacity;
   int* harr;
   };
 struct MinHeap* createMinHeap(unsigned capacity)
 {
 struct MinHeap* minHeap = new MinHeap;
 minHeap->size = 0;
 minHeap->capacity = capacity;
 minHeap->harr = new int[capacity];
 return minHeap;
}
void swapMinHeapNode(int* a, int* b)
 {
   int temp = *a;
   *a = *b;
   *b = temp;
```

```
}
void minHeapify(struct MinHeap* minHeap, int idx)
 {
  int smallest = idx;
  int left = 2 * idx + 1;
  int right = 2 * idx + 2;
  if (left < minHeap->size
          && minHeap->harr[left] < minHeap->harr[smallest])
           smallest = left;
    if (right < minHeap->size
          && minHeap->harr[right] < minHeap->harr[smallest])
          smallest = right;
        if (smallest != idx) {
swapMinHeapNode(&minHeap->harr[smallest],&minHeap->harr[idx]);
minHeapify(minHeap, smallest);
    }
}
 int isSizeOne(struct MinHeap* minHeap)
    {
      return (minHeap->size == 1);
  }
 int extractMin(struct MinHeap* minHeap)
{
   int temp = minHeap->harr[0];
   minHeap->harr[0] = minHeap->harr[minHeap->size - 1];
  --minHeap->size;
  minHeapify(minHeap, 0);
   return temp;
}
void insertMinHeap(struct MinHeap* minHeap, int val)
{
   ++minHeap->size;
   int i = minHeap->size - 1;
   while (i && (val < minHeap->harr[(i - 1) / 2])) {
  minHeap->harr[i] = minHeap->harr[(i - 1) / 2];
  i = (i - 1) / 2;
}
minHeap->harr[i] = val;
}
void buildMinHeap(struct MinHeap* minHeap)
{
   int n = minHeap->size - 1;
    int i;
     for (i = (n - 1) / 2; i >= 0; --i)
     minHeapify(minHeap, i);
 }
struct MinHeap* createAndBuildMinHeap(
      int len[], int size)
{
       struct MinHeap* minHeap = createMinHeap(size);
       for (int i = 0; i < size; ++i)
       minHeap->harr[i] = len[i];
       minHeap->size = size;
      buildMinHeap(minHeap);
      return minHeap;
}
int minCost(int len[], int n)
{
 int cost = 0;
 struct MinHeap* minHeap = createAndBuildMinHeap(len, n);
 while (!isSizeOne(minHeap)) {
               int min = extractMin(minHeap);
               int sec_min = extractMin(minHeap);
            cost += (min + sec_min);
            insertMinHeap(minHeap, min + sec_min);
        }
          return cost;
}
      int main()
    {
       int n;
       cin >> n;
       int arr[n];
       for(int i = 0; i < n; i++)
       cin >> arr[i];
       int size = sizeof(arr) / sizeof(arr[0]);
       cout << minCost(arr, size);
       return 0;
}
```

Q3) A party has been organised on a cruise. The party is organised for a limited time(T). The number of guests entering (E[i]) and leaving (L[i]) the party at every hour is represented as elements of the array. The task is to find the maximum number of guests present on the cruise at any given instance within T hours.

Example 1:

Input:
5 ---> Value of T
[7,0,5,1,3] ---> E[], element of E[0] to E[N-1], where input each element is separated by new line
[1,2,1,3,4] -----> L[],element of L[0] to L[N-1], where input each element is separated by new line

Output:
8 -----> Maximum number of guests on cruise at an instance.

Explanation:

1st hour Entry: 7, Exit: 1
No. of guests on the ship: 6
2nd hour:
1st hour
Entry: 0, Exit: 2
No. of guests on the ship: 6 -2 = 4
Hour 3: Entry: 5, Exit: 1
No. of guests on the ship: 4 + 5 -1 = 8
Hour 4:
Entry: 1, Exit: 3
No. of guests on the ship: 8 + 1 - 3 = 6
Hour 5:
Entry: 3, Exit: 4
No. of guests on the ship: 6 + 3 - 4 = 5
Hence, Maximum Number of guests within 5 hours is 8.

Code Solution:

```
#include <iostream>
using namespace std;
int main()
{
 int n, sum = 0,
 max; cin >> n;
 int a[n], b[n];
 for(int i = 0; i < n; i++)
{
 cin >> a[i];
}
for(int i = 0; i < n; i++)
 {
 cin >> b[i];
}
max = sum;
for(int i = 0; i < n; i++)
{
 sum = sum + a[i] - b[i];
 if(max < sum)
 max = sum;
}
cout << max;
}
```

Q4) Given an array Arr[] of size T, contains binary digits.
Where
0 represents a biker running to the north.
1 represents a biker running to the south.
The task is to count crossing bikers in such a way that each pair of crossing bikers (N, S), where 0<=N<S<T,
is passing when N is running to the north and S is running to the south.
Constraints:
<=N<S<T
Example -1:

Input:

5.-> Number of elements i.e. T

0.-> Value of 1st element
1.-> Value of 2nd element
0.-> Value of 3rd element
1.-> Value of 4th element
1.-> Value of 5th element
Output:
5

Explanation:

The 5 pairs are (Arr[0], Arr[1]), (Arr[0], Arr[3]), (Arr[0], Arr[4]), (Arr[2], Arr[3]) and (Arr[2], Arr[4]).

Note that in all pairs first element is 0, second element is 1 and index of first element is smaller than index of second element.
The Input format for testing:
First input line: Accept a single positive integer value for T representing the size of Arr[]. Second input line:: Accept N number of integer values (0 or 1) separated by a new line.

Output Format for Testing:

The output must be a non-negative integer number only (See the output format in example). Additional messages in the output will result in the failure of test cases.

Code Solution:

```
#include <iostream>
using namespace std;
int main()
{
 int n, count=0;
  cin>>n;
int B[n];
for(int i=0; i<n; i++)
cin>>B[i];
for(int i=0; i<n; i++)
{
if(B[i]==0)
{
for(int k=i+1; k<n; k++)
{
if(B[k]==1)
count++;
}
}
}
cout<<count;
}
```

Q5)A supermarket maintains a pricing format for all its products. A value N printed on each product. When the scanner reads the value N on the item, the product of all the digits in the value N is the price of the item. The task is to design a software such that given the code of any item N the product(multiplication) of all the digits of value should be computed(price).

Example 1:
Input:
5244 -->Value of N
Output:
160 -->Price

Explanation:
From the input above:
Product of the digits: 5,1,4,4
5*2*4*4 = 160
Hence Output is 160

Code Solution:

```
#include <iostream>
using namespace std;
int main()
{
  int n, rem, mul = 1;
  cin >> n;
  while(n!=0)
   {
        rem = n%10;
        mul = mul * rem;
        n = n/10;
    }
cout << mul;
}
```

Q6)An event management company has come up with a unique idea of printing their event tickets. Based on the ticket number combination (str1), the visitor is directed towards a particular class of audience. The task is to create a program/application to fetch the ticket number based on the following conditions: Any

occurrences of digits EF, 56 and G, & should be deleted The characters EF should be in the same format.

Example 1:
Input:
4523EF58G -> Value of STR1

Output:
452358 -> After removal of characters 'EF' and 'G'

Example 2:
Input:
E12F35G58 -> Value of STR1

Output:
E12F3558 -> After removal of character 'G'

Explanation:

In the above example, characters E and F are not together. So, they won't be deleted. The output will be with only character G removal.

The Input format for testing The candidate has to write the code to accept 1 input(s). First input - Accept value for str1 which is a string consisting of numbers and uppercase alphabets without any spaces.

The output format for testing The output should be a string without any spaces (Check the output in Example 1 and Example

2) Additional messages in output will cause the failure of test cases.

Constraints:
Str={(A,Z),(0-9)}
No spaces and special characters allowed.
Only uppercase alphabets in the input string

Code Solution:

```
#include<iostream>
 #include<string.h>
 using namespace std;
 int main()
{
string T, F;
cin>>T;
int len = T.length();
int j=0;
for(int i=0; i<len; i++)
{
if((T[i]=='E'&&T[i+1]=='F') || (T[i]=='5'&&T[i+1]=='6'))
   {
    i++;
    continue;
  }
   else if(T[i]=='G' || T[i]=='7')
  {
    continue;
  }
   else
   {
```

```
 F[j]=T[i];
 j++;
}
}
for(int i=0; i<j; i++)
cout<<F[i];
}
```

Q7) A carpet manufacturing industry has newly ventured into the carpet installation business. All the carpets manufactured are large squares in shape. To install, each carpet has to be cut into shapes of squares or rectangles. The number of slits to be made is given as N. The task is to find the maximum number of equal squares or rectangles that can be achieved using N slits.

Note:

The square carpet can be cut only using horizontal or vertical slits. Cuttings are done on a single carpet which should be rolled out completely i.e. no folding or stacking is allowed.
Squares or rectangles cut should be equal size.
Example 1:
Input:
4 → Value of N(No. of cuts)

Output:
9 → maximum number of equal squares or rectangles

Explanation:
Solution 2

Maximum number of squares/rectangles that can be obtained with N=4 is 9(Solution 1)
Hence, output is 9
Example 2:
Input:
1 → Value of N(No. of teams)
Output:
2 → maximum number of equal squares or rectangles

Code Solution:

```
#include <bits/stdc++.h>
using namespace std;
int findMaximumPieces(int n)
{
int x = n / 2;
return ((x + 1) * (n - x + 1));
}
int main()
{
int n;
cin>>n;
cout <<findMaximumPieces(n);
return 0;
}
```

Q8) A family is about to break their piggy bank to use the money for different purposes. The piggy bank here represents an array (arr[]) consisting of N coins. The family has to split the coins of piggy bank into smaller stack (sub-array) of coins such that the sum of the difference between the maximum value and the minimum value of the coins for all the stacks (sub-arrays) is maximum.

Note: Each value of the array can be used only once that is only in one subarray.

Constraints:
1 <= N <= 500
1 <=arr[i] <= 100
Example 1:
Input:
5 → Value of N
{8,1,7,9,2} → arr[] elements from arr[0] to arr [N-1],
Where each element is separated by new line.
Output:
14
Explanation:
Let us break the array elements into following subarrays:
1. (8,1) → Max:8 Min:1
2. (7,9,2) → Max:9 Min:2
So, the difference between the maximum and minimum elements in each subarrays is
1. 8-1=7
2.9-2=7

Now, the sum of the differences of subarray is: 7+7=14
Hence, output is 14.

Example 2:
Input:
5 → Value of N
{1,2,1,0,5} → arr[], elements from arr[0] to arr [N-1], where each elements is separated by a new line.
Output:
6
Explanation:
Let us break the array elements into following subarrays:
1. (1,2,1) → max:2, min:1
2. (0,5) → max:5, min:0

So, the difference between the max and min elements in each subarray is
1. 2-1 = 1
2. 5-0 = 5

Now, the sum of the differences of subarray is:
1+5 = 6
Hence, output is 6.

The input format for testing The candidate has to write the code to accept 2 inputs.
First input - Accept value for N(positive integer number)
Second input - Accept N number of values (arr[]), where each value is separated by a new line.

Code solution:

```
#include<iostream>
using namespace std;
int main()
```

```
 {

   int n, sum = 0;
   cin >> n;
   int arr[n];
 for(int i = 0; i < n; i++)
   {
  cin >> arr[i];
  }
  for(int i = 0; i < n; i++)
   {
    for(int j = i+1; j < n; j++)
{
        if(arr[i] < arr[j])
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
    }
   }
}

for(int i = 0; i < n/2; i++)
 {
  sum = sum+(arr[i]-arr[n-i-1]);
}
cout << sum;
}
```

Q9) Given a non-negative integer array Arr having size N. Each element of the array will carry a different value. This means no

two elements can have the same values.The candidate has to do this with minimal changes in the original value of the elements, making every element as least as much value as it originally had. Find the minimum sum of all elements that can be set the array for:

Example 1:

Input
3 -> Value of N, represents size of Arr
2 -> Value of Arr[0]
2-> Value of Arr[1]
4-> Value of Arr[2]

Output
9

Explanation:

As two elements have the same value, max value for the one of them needs to be incremented to 3.
He can set the array with 2+3+4=9

Example 2:
Input
2 -> Value of N, represents size of Arr
3 -> Value of Arr[0]
4-> Value of Arr[1]
5-> Value of Arr[2]

Output
Wrong Input

Explanation:
Here N=2, so we need to provide value of only two elements but we are providing value of three elements so result is "Wrong Input"

The Input format for the testing

First input line: Accept a single positive integer value for N representing the size of Arr[]
Second input line: Accept N number of integer values separated by a new line, representing the original value assigned to each element.
Output Format for testing:
The output must be a non integer only (See the output format example).

Code Solution:

```
#include<iostream>
using namespace std;
int main()
{
  int n, sum = 0, flag = 0;
  cin >> n;
  int arr[n];
    for(int i = 0; i < n; i++)
     {
      cin >> arr[i];
   }
for(int i = 0; i < n; i++)
   {
       for(int j = i+1; j < n; j++)
        {
            if(arr[i] == arr[j])
                  {
                     arr[i] = arr[i]+1;
              }
             if(arr[j] < 0)
             flag = 1;
           }
       }
     for(int i = 0; i < n; i++)
        {
         sum = sum + arr[i];
       }
          if(flag == 1)
          cout << "Wrong Input";
          else
          cout << sum;
    }
```

Q10) Joseph is learning digital logic subject which will be for his next semester. He usually tries to solve unit assignment problems before the lecture. Today, he got one tricky question. The problem statement is "A positive integer has been given as an input. Convert decimal value to binary representation. Toggle all bits of it after the most significant bit including the most significant bit. Print the positive integer value after toggling all bits".

Constraints
1 <=N <=100

Example 1
Input:
10 ---> Integer

Output:
5 → result - Integer

Explanation:
Binary representation of 10 is 1010. After toggling the bits (1010), will get 0101, which represents “5”.
Hence the output will print “5”.

Example 2
Input:
101 ---> Integer
Output:
Wrong input → result - String

Explanation:
Given integer “101” is out of range. Hence the output will print “Wrong input”.

The input format for testing The candidate has to write the code to accept one input(s).
1. First Input - First line contains an integer

The output format for testing

1. Print integer value based on the number got after toggling all the bits of given input.
2. Print “Wrong input if the string length is out of the range.
3. Additional messages in output will cause the failure of test cases

Instructions
1. The system doesn’t allow any kind of hard-coded input value.
2. Written program code by the candidate will be verified against all the inputs which are supplied from the system

Code Solution:

```
#include<iostream>
 using namespace std;
void toggle(int &n)
{

   int temp = 1;
   while (temp <= n)
        {
            n = n ^ temp;
             temp = temp << 1;
          }
        }
    int main()
   {
       int n;
       cin >> n;
      if(n <= 100)
       {
            toggle(n);
             cout << n;
      }
        else
        cout << "Wrong Input";
       return 0;
}
```

Q11) Find sub arrays with a given sum in array.Given integer array find subarrays with a given sum in it.

Input :- arr = [3,4,-7,1,3,3,1,-4]
Target = 7

Output:
[3,4]
[3,4,-7,1,3,3]
[1,3,3]

[3,3,1]

Code :-

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

void optimalApproach(vector<int>& arr, int n, int target) {
    unordered_map<int, int> sumMap;
    int curSum = 0;
    for (int i = 0; i < n; i++) {
        curSum += arr[i];
        if (curSum == target) {
            for (int j = 0; j <= i; j++) {
                cout << arr[j] << " ";
            }
            cout << endl;
        }
        if (sumMap.find(curSum - target) != sumMap.end()) {
            int startIndex = sumMap[curSum - target] + 1;
            for (int k = startIndex; k <= i; k++) {
                cout << arr[k] << " ";
            }
            cout << endl;
        }
        sumMap[curSum] = i;
    }
}

int main() {
    vector<int> arr = {3, 4, -7, 1, 3, 3, 1, -4};
    int N = arr.size();
    int target = 7;
    optimalApproach(arr, N, target);
    return 0;
}
```

Q12) There is a robot on an m x n grid. The robot is initially located at the top-left corner (ie, grid[0] [0]). The robot tries to move to the bottom-right corner (i.e, grid[m- 1][n-1]) The robot can only move either down or right at any point in time.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.

The test cases are generated so that the answer will be less than or equal to 2=10^9

Code:-

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

class Solution {
private:
    int f(int i, int j, vector<vector<int>> &dp){
        if(i == 0 && j == 0) return 1;
        if(i < 0 || j < 0) return 0;
        if(dp[i][j] != -1) return dp[i][j];

        int up = f(i-1, j, dp);
        int left = f(i, j-1, dp);
        return dp[i][j] = up + left;
    }
public:
    int uniquePaths(int m, int n) {
        vector<vector<int>> dp (m, vector<int>(n, -1));
        return f(m-1, n-1, dp);
    }
};
```

Q13) Given two Integer, find sum of cubes all numbers in the range of n&m .

Input n= 4 ,m = 9

Output = 1989

Code:-

```
#include <iostream>

using namespace std;

void findCubeSum(int start, int end) {
    int cubeSum = 0;
    for (int i = start; i <= end; i++) {
        cubeSum += i * i * i;
    }
    cout << cubeSum << endl;
}

int main() {
    int start, end;
    cin >> start >> end;
    findCubeSum(start, end);
    return 0;
}
```

Q14)You are given a grocery list which consists of three parameters

Item, quanting, price

Your task is to find

→ Higher selling item

→Total Selling item

→selling item item

Input :
Apple[1.0,5]
Orange[10.0,5]
Apple[10.0,5]

Output:
Apple
105
35.0

Code:-

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

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

    // Method 1
    int maxCost = 0;
    string maxCostItem = "";
    double totalPrice = 0;
    double avg = 0;

    for (int i = 0; i < n; i++) {
        string item;
        int quantity, price;
        cin >> item >> quantity >> price;

        totalPrice += quantity * price;
        if (totalPrice > maxCost) {
            maxCostItem = item;
        }
        avg = totalPrice / (i + 1);
    }
    cout << "Task 1 - Item: " << maxCostItem << "\nTotal price: " <<
fixed << totalPrice << "\nAverage Price: " << avg << endl;

    // Method 2
    map<string, double> store;
    maxCost = 0;
    maxCostItem = "";
    totalPrice = 0;

    for (int i = 0; i < n; i++) {
        string item;
        int quantity, price;
        cin >> item >> quantity >> price;
        totalPrice = quantity * price;
        store[item] += totalPrice;

        if (maxCost < store[item]) {
            maxCostItem = item;
            maxCost = store[item];
        }
    }

    double total = 0;
    for (auto& pair : store) {
        total += pair.second;
    }
    avg = total / n;

    cout << "\nTask 2 - Item: " << maxCostItem << "\nTotal price: " <<
fixed << total << "\nAverage Price: " << avg << endl;

    return 0;
```

```
}
```

Q15) Given an Integer, we need to find the sum of values of that table.

Input: 10
output: 550

Code:

```
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    int mSum = 0;
    for (int i = 1; i <= n; i++) {
        mSum += n * i;
    }
    cout << mSum << endl;
    return 0;
}
```

Q16) Given an array and a integer K. we need to find the maximum dement in each of the contiguous subarrays.

Input -247163

K=3

Output -7776

Code:-

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

using namespace std;

void optimalSolution(vector<int>& arr, int k) {
    priority_queue<pair<int, int>> maxHeap;
    vector<int> ans;

    for (int i = 0; i < k; i++) {
        maxHeap.push({arr[i], i});
    }
    ans.push_back(maxHeap.top().first);

    for (int i = k; i < arr.size(); i++) {
        maxHeap.push({arr[i], i});
        while (maxHeap.top().second <= i - k) {
            maxHeap.pop();
        }
        ans.push_back(maxHeap.top().first);
    }
    for (int num : ans) {
        cout << num << " ";
    }
    cout << endl;
}

int main() {
    vector<int> arr;
    int num;
    while (cin >> num) {
        arr.push_back(num);
        if (cin.get() == '\n') break;
    }
    int k;
    cin >> k;
    solve(arr, k);
    optimalSolution(arr, k);
    return 0;
}
```

Q17) Calculate the Sum of N terms of Fibonacci Series

Note,

fib(0)=1

fib(1) = 1

Input
n=5

Output : 7

Code:-

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

int fibonacci(int n) {
    int a = 0, b = 1, sum=1;
    if (n < 0) {
        cout << "Incorrect input" << endl;
        return -1; // or any error code you prefer
    } else if (n == 0) {
        return a;
    } else if (n == 1) {
        return b;
    } else {
        for (int i = 2; i < n; ++i) {
            int c = a + b;
            sum += c;
            a = b;
            b = c;
        }
        return sum;
    }
}

int main() {
    int n;
    cin >> n;
    cout << fibonacci(n) << endl;
    return 0;
}
```

Q18) Bitwise ORs of Subarrays

Given an Integer array, return the number of distinct bitwise ORs of all the non-empty subarrays of arr.

The bitwise OR of a subarray is the bitwise OR of each integer in the subarray. The bitwise OR of a subarray of one integer is that integer.

A subarray is a contiguous non empty sequence of elements within an array.

Example 1:

Inputs arr[0]

Output: 1

Explanation: There is only one possible result: 0.

Example 2:

Inputs arr [1,1,2]

Output: 3

Explanations :The possible subarrays are [1], [1], [2], [1, 1], [1,2],[1,1,2]

These yield the results 1, 1, 2, 1, 3, 3.

There are 3 unique values, so the answer is 3.

Example 3

Inputs arr [1,2,4]

Output 6

Explanations The possible results are 1, 2, 3, 4, 6, and 7

Code :

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

int subarrayBitwiseORs(vector<int>& arr) {
    vector<int> res;
    int left = 0;
    for (int num : arr) {
        int right = res.size();
        res.push_back(num);
        for (int i = left; i < right; ++i) {
            int value = res[i] | num;
            if (res.back() != value) {
                res.push_back(value);
            }
        }
        left = right;
    }
    return unordered_set<int>(res.begin(), res.end()).size();
}
```

```
int main() {
    int n;
    cin >> n;
    vector<int> arr(n);
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }
    cout << subarrayBitwiseORs(arr) << endl;
    return 0;
}
```

Q19) In a database there are N students, the fields of the table are name,age ,grade,gender . Your task is to return the students name who are greater than 20 years old & calculate the average of grades using ascii values of female candidates .

Input : 3

```
        AAA    21    A    Female
        BBB    21    B    Male
        CCC    24    C    Female
```

Output -

```
        AAA    BBB    CCC
        66
```

Code:

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

int main() {
    int n;
    vector<string> ans;
    int totalFemale = 0, totalGrade = 0;
    while(n--) {
        string name, gender;
        char grade;
        int age;
        cin >> name >> age >> grade >> gender;

        if(age > 20)
            cout << name << endl;

        if (gender == "Female") {
            totalFemale++;
            totalGrade += grade;
        }
    }
    double avg = (double) (totalGrade / totalFemale);
    cout << avg << endl;

}
```

Q20) find Majority Elements, Majority elements is the element which occurs more than or equal to 1/2 times the array, where 'N' is the size of the array.

Input- 6
[2 4 2 4 2 4]
Output - 2,4

Code-

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

void findMajorityElement(vector<int>& arr, int N) {
    unordered_map<int, int> freq;
    for (int num : arr) {
        freq[num]++;
    }
    for (auto& pair : freq) {
        if (pair.second >= N / 2) {
            cout << pair.first << " ";
        }
    }
}

int main() {
    int N;
    cin >> N;
    vector<int> array(N);
    for (int i = 0; i < N; ++i) {
        cin >> array[i];
    }
    findMajorityElement(array, N);
    return 0;
}
```

Q21) the organization has data warehouse there will be given a three digit number. Check whether number is divisible by 9 or not?

Input -236

o/p- Number 236 is not divisible by 9

Input -162
Number 162 is divisible by 9.

Code:

```
#include <iostream>

bool isDivisibleByNine(int number) {
    return number % 9 == 0;
}

int main() {
    int number;
    std::cin >> number;
    if (number >= 100 && number <= 999) {
        if (isDivisibleByNine(number)) {
            std::cout << "Number " << number << " is divisible by 9" <<
std::endl;
        } else {
            std::cout << "Number " << number << " is not divisible by 9" <<
std::endl;
        }
    }

    return 0;
}
```

Q22) we are given a lIst of numbers we need to return maximum difference b/w Smallest & Largest Number.

Input- smallest number should be before largest number.

Ex , n= 7
[-3, -5, 1, 6, -7, 8, 11]

ans - 18

Code:

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

using namespace std;

int solve(vector<string>& arr) {
    int largest = INT_MIN;
    int maxDiff = INT_MIN;
    for (int i = arr.size() - 1; i >= 0; --i) {
        try {
            int num = stoi(arr[i]);
            largest = max(largest, num);
            maxDiff = max(maxDiff, largest - num);
        } catch (const std::invalid_argument& e) {
            // Handle invalid input
            std::cerr << "Invalid input: " << arr[i] << std::endl;
        }
    }
    return maxDiff;
}
```

```
int main() {
    // Input array format
    cout << "Enter elements separated by commas within square
brackets like [element1,element2,...]: ";
    string input;
    getline(cin, input);
    input.erase(remove(input.begin(), input.end(), '['), input.end()); //
Remove '['
    input.erase(remove(input.begin(), input.end(), ']'), input.end()); //
Remove ']'
    stringstream ss(input);
    vector<string> arr1;
    string element;
    while (getline(ss, element, ',')) {
        arr1.push_back(element);
    }
    cout << "Case 1 - [-3,-5,1,6,-7,8,11], maxdiff: " << solve(arr1) <<
endl;

    // Input space separated
    cout << "Enter elements separated by spaces: ";
    string spaceSeparatedInput;
    getline(cin, spaceSeparatedInput);
    vector<int> arr2;
    int num;
    stringstream ss2(spaceSeparatedInput);
    while (ss2 >> num) {
        arr2.push_back(num);
    }
    cout << "Case 2: -3 -5 1 6 -7 8 11, maxdiff: " << solveInt(arr2) <<
endl;
```

```
    return 0;
}
```

Q23) A Person has many shoes of different sizes and he wants to arrange them, Calculate the number of pairs of shoes.

Example 1:

8

7L 7R 7L 8L 6R 7R 8R 6R

O/P: 3

Example 2:

5

7R 7L 8R 10R 10L

O/P: 2

Code:-

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

int main() {
    int N;
    cin >> N;
    unordered_map<string, int> store;

    for (int i = 0; i < N; ++i) {
        string shoeSize;
        cin >> shoeSize;
        if (store.find(shoeSize) != store.end()) {
            store[shoeSize]++;
        } else {
            store[shoeSize] = 1;
        }
    }

    int pairs = 0;
    for (auto& it : store) {
        string key = it.first;
        int val = it.second;
        string opposite;
        if (key[key.size() - 1] == 'L') {
            opposite = key.substr(0, key.size() - 1) + 'R';
        } else {
            opposite = key.substr(0, key.size() - 1) + 'L';
        }
        if (store.find(opposite) != store.end()) {
            pairs += min(val, store[opposite]);
        }
    }

    cout << pairs / 2 << endl;

    return 0;
}
```

Q24) in a company there are employees and their efficiency is given in array 'arr' (can be negative) you need to find the maximum efficiency of 3 employees. The efficiency of 3 employees will be calculated by multiplying their individual efficiencies from the given array.

Example 1:

5

[3 -2 -8 4 1]

O/P: 64

Code:

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

int main() {
    int N;
    cin >> N;
    vector<int> arr(N);
    for (int i = 0; i < N; ++i) {
        cin >> arr[i];
    }
    sort(arr.begin(), arr.end());
    // Max values
    int emp1 = arr[N - 1];
    int emp2 = arr[N - 2];
    int emp3 = arr[N - 3];

    // Negative case
    int neg1 = arr[0];
    int neg2 = arr[1];

    cout << max(emp1 * emp2 * emp3, neg1 * neg2 * emp1) << endl;

    return 0;
}
```

Q25) Given an array nums of size n, return the majority element. The majority element is the element that appears more than [n/3] times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [3,2,3,3]

Output: 3

Example 2:

Input: nums = [2,2,1,0,1,2,2,2,2]

Output: 2

Code:

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

using namespace std;

void findMajorityElement(vector<string>& arr, int N) {
    unordered_map<string, int> freq;
    for (string& n : arr) {
        if (freq.find(n) != freq.end()) {
            freq[n]++;
        } else {
            freq[n] = 1;
        }
    }
    for (auto& it : freq) {
        if (it.second >= N / 3) {
            cout << it.first << " ";
        }
    }
}

void findMajorityElementInt(vector<int>& arr, int N) {
    unordered_map<int, int> freq;
    for (int& num : arr) {
        if (freq.find(num) != freq.end()) {
            freq[num]++;
        } else {
            freq[num] = 1;
        }
    }
    for (auto& it : freq) {
        if (it.second >= N / 3) {
            cout << it.first << " ";
        }
    }
}

void input_array_format() {
    cout << "\nEnter elements in format [e1,e2,e3]: ";
    string input;
    getline(cin, input);
    stringstream ss(input);
    string token;
    vector<string> arr;
    while (getline(ss, token, ',')) {
        arr.push_back(token);
    }
    findMajorityElement(arr, arr.size());
}

void input_space_separated() {
    cout << "\nCase 2: Enter space-separated elements: ";
    string input;
    getline(cin, input);
    stringstream ss(input);
    int num;
    vector<int> arr;
    while (ss >> num) {
        arr.push_back(num);
    }
    findMajorityElementInt(arr, arr.size());
```

```
}

int main() {
    input_array_format();
    input_space_separated();
    return 0;
}
```

Q26) Given an array nums with n objects colored red, white, or blue, sort them

in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 3, 6 and 7 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

Input: nums = [3,6,3,7,6,3,7]

Output: [3 3 3 6 6 7 7]

Code:-

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

using namespace std;
```

```
vector<char> sortBySwap(vector<char>& arr, int N) {
    int left = 0;
    int mid = 0;
    int right = N - 1;
    while (mid <= right) {
        if (arr[mid] == '3') {
            swap(arr[left], arr[mid]);
            left++;
            mid++;
        } else if (arr[mid] == '6') {
            mid++;
        } else if (arr[mid] == '7') {
            swap(arr[right], arr[mid]);
            right--;
        }
    }
    return arr;
}

int main() {
    cout << "\nEnter space-separated elements: ";
    vector<char> arr;
    char c;
    while (cin >> c) {
        arr.push_back(c);
    }
    arr = sortBySwap(arr, arr.size());
    for (char c : arr) {
        cout << c << " ";
    }
    return 0;
}
```

Q27) Given an array of size N-1 with integers in the range of [1,N], the task is to find the missing number from the first N integers.

Example 1:

Input:

4

[1 2 3 5]

Output: 4

Example 2:

Input:

3

[1 2 4]

Output: 3

Code:

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

using namespace std;
```

```
int getMissingNumIntFormat(vector<int>& arr, int size) {
    int totalSum = 0;
    int n = size + 1;
    for (int num : arr) {
        totalSum += num;
    }
    int actualSum = (n * (n + 1)) / 2;
    return actualSum - totalSum;
}

int getMissingNumStrFormat(vector<string>& arr, int size) {
    int totalSum = 0;
    int n = size + 1;
    for (auto str : arr) {
        int num = stoi(str); // Convert string to integer using stoi
        totalSum += num;
    }
    int actualSum = (n * (n + 1)) / 2;
    return actualSum - totalSum;
}



void input_space_separated() {
    cout << "\nEnter space-separated elements: ";
    int N;
    cin >> N;
    cin.ignore();
    string input;
    getline(cin, input);
    stringstream ss(input);
    int num;
    vector<int> arr;
    while (ss >> num) {
        arr.push_back(num);
    }
    cout << "Missing number: " << getMissingNumIntFormat(arr, N) <<
endl;
}


void input_array_format() {
    cout << "\nEnter elements in format e1,e2,e3: ";
    string input;
    int N; cin >> N;
    cin.ignore();
    getline(cin, input);
    stringstream ss(input);
    string token;
    vector<string> arr;
    while (getline(ss, token, ',')) {
        arr.push_back(token);
    }
    cout << "Missing number: " << getMissingNumStrFormat(arr, N) <<
endl;
}

int main() {
    input_space_separated();
    input_array_format();
    return 0;
}
```

Q28) Write a program that accepts two integers, nn and mm, and prints all prime numbers between nn and mm (inclusive) such that the sum of their digits is also a prime number.

Example 1:

Input: [20 25]

Output: 23

Code:

```
#include <iostream>

using namespace std;

bool isPrime(int number) {
    if (number <= 1) {  // 1 and below are not prime
        return false;
    }
    if (number <= 3) {  // 2 and 3 are prime
        return true;
    }
    if (number % 2 == 0 || number % 3 == 0) {  // Eliminate multiples of
2 and 3
        return false;
    }

    // Check divisibility by numbers of the form 6k ± 1, up to sqrt(n)
    int i = 5;
    while (i * i <= number) {
        if (number % i == 0 || number % (i + 2) == 0) {
            return false;
        }
        i += 6;
    }
    return true;
}

bool calculateSum(int n) {
    int sum = 0;
    while (n > 0) {
        int num = n % 10;
        sum += num;
        n /= 10;
    }
    return isPrime(sum);
}

int main() {
    int n, m;
    cin >> n >> m;
    for (int i = n; i <= m; i++) {
        if (isPrime(i) && calculateSum(i)) {
            cout << i << endl;
        }
    }
    return 0;
}
```

Q29) Write a program to take input of X and Y in a new line. Print the number which is nearer the integer when divided by Y.

Example 1:

Input:

X=13

Y-3

Output: 12

Example 2:

Input:

X=13

Y=3

Output: 12

Code:

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

int customRound(double number) {
    int integerPart = (int) number;
    double decimalPart = number - integerPart;
    if (decimalPart >= 0.5) {
        return integerPart + 1;
    } else {
        return integerPart;
    }
}

int main() {
    int x, y;
    cin >> x;
    cin >> y;

    double nearest = (double) x / y;
    int roundedNumber = customRound(nearest);
    cout << roundedNumber * y << endl;

    return 0;
}
```

Q30) Write a program that generates a password adhering to the following conditions: • The password must consist of at least 8 characters.

• It must contain at least one integer.

• It must contain at least one special character from the set ('#', '@'). • It must contain at least one uppercase letter and one lowercase letter.

• Each character in the password should be incremented by the number of times specified by the second input.

Your program should take two inputs:

1.A string representing the initial password.

2. An integer representing the number of times each character should be incremented. Your program should then generate and output the modified password.

Example 1:

Input1: werV432@

Input2:2

Output: ygtX653#

Code:

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

string addValueToChars(string inputStr, int value) {
    string result = "";
    for (char ch : inputStr) {
        if (isalpha(ch)) {
            if (islower(ch)) {
                result += ((ch - 'a' + value) % 26) + 'a';
            } else {
                result += ((ch - 'A' + value) % 26) + 'A';
            }
        } else if (isdigit(ch)) {
            result += ((ch - '0' + value) % 10) + '0';
        } else {
            if (ch == '@') {
                result += '#';
            } else {
                result += '@';
            }
        }
    }
    return result;
}

int main() {
    string inputStr;
    int value;

    cout << "Enter input string: ";
    cin >> inputStr;
    cout << "Enter value: ";
    cin >> value;

    string outputStr = addValueToChars(inputStr, value);
    cout << "Output: " << outputStr << endl;

    return 0;
}
```

Q31) given size of n and list of array elements and we should print if the given element in array is divisible by 3 then replace the element with "Three" and if the element in array is divisible by 5 then replace the element with "Five" if the element divisible

by 3 and 5 both then replace the element with "ThreeFive" if the element in the array is not satisfying the above 3 conditions then put the element as it is and print the array

Example 1:

Input:

N=4

[2 3 4 5]

Output: 2 Three 4 Five

Example 2:

Input:

N=2

[353]

Output: Three Five Three

Code:

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

using namespace std;

void solve(vector<string>& arr) {
    stringstream result;
    for (auto n : arr) {
        int num = stoi(n);
        if (num % 3 == 0 && num % 5 == 0) {
            cout << "ThreeFive " << " ";
        } else if (num % 5 == 0) {
            cout << "Five "<< " ";
        } else if (num % 3 == 0) {
            cout << "Three "<< " ";
        } else {
            cout << num << " ";
        }
    }
}

void input_array_format1() {
    int n; cin >> n;
    cin.ignore();
    string input;
    getline(cin, input);
    input.erase(remove(input.begin(), input.end(), '['), input.end()); //
Remove '['
    input.erase(remove(input.begin(), input.end(), ']'), input.end()); //
Remove ']'
    stringstream ss(input);
    vector<string> arr1;
    string element;
    while (ss >> element) { // Read elements separated by space
        arr1.push_back(element);
    }
    solve(arr1); // Print the result
}
```

```
int main() {
    input_array_format1();
    return 0;
}
```

Q32) Task 1- print the collatz sequence upto ending with 1 the sequence should be in the following way

F(n)=n//2

> if the number is even:

> if the number is odd:

F(n)=3*n+1

This sequence should end until the last element of sequence is 1

Task-2

For the given integer from 1,n it should calculate the sequence of each k value I.e, 1<=k<=n Calculate the maximum length of sequence list of each k value and return the maximum length of the sequence list of the k value

Task-3

Calculate the maximum value of the each sequence within the sequence list of each k value and return the maximum value of the sequence list of the k value and those k value itself

and the k value itself For the given integer from 1.n it should calculate the sequence of each k value Ie, 1<=k<=n

Input: 5

Output:

[5,16,8,4,2.1]

8,3

16,3

Input: 0

Output: Error!

Input: xyza

output: Error!

Input:-13 Output: Error!

Input: 5.5

Output: Error!

Code:

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

```
using namespace std;

bool is_positive_integer(const string& str) {
    for (char c : str) {
        if (!isdigit(c)) {
            return false;
        }
    }
    return !str.empty() && stoi(str) > 0;
}

vector<int> generate_sequence(int n) {
    vector<int> sequence;
    sequence.push_back(n);
    while (n != 1) {
        if (n % 2 == 0)
            n /= 2;
        else
            n = 3 * n + 1;
        sequence.push_back(n);
    }
    return sequence;
}

pair<int, int> max_length_of_sequence_and_k(int n) {
    int maxLen = 0, kValue = 0;
    for (int i = 1; i <= n; ++i) {
        vector<int> sequence = generate_sequence(i);
        int currLen = sequence.size();
        if (currLen > maxLen) {
            maxLen = currLen;
            kValue = i;
        }
    }
    return make_pair(maxLen, kValue);
}

pair<int, int> max_sequence_value_and_k(int n) {
    int max_k = 0, max_val = 0;
    for (int k = 1; k <= n; ++k) {
        vector<int> sequence = generate_sequence(k);
        int max_val_k = *max_element(sequence.begin(),
sequence.end());
        if (max_val_k > max_val) {
            max_val = max_val_k;
            max_k = k;
        }
    }
    return make_pair(max_val, max_k);
}

int main() {
    string n_str;
    cin >> n_str;
    if (is_positive_integer(n_str)) {
        int n = stoi(n_str);
        vector<int> sequence = generate_sequence(n);
        auto max_val_k = max_sequence_value_and_k(n);
        auto maxLen_kValue = max_length_of_sequence_and_k(n);
        cout << "Sequence: [";
        for (size_t i = 0; i < sequence.size(); ++i) {
            cout << sequence[i];
            if (i != sequence.size() - 1) {
                cout << ", ";
```

```
          }
        }
        cout << "]" << endl;
        cout << maxLen_kValue.first << " " << maxLen_kValue.second
<< endl;
        cout << max_val_k.first << " " << max_val_k.second << endl;
    } else {
        cout << "Error!" << endl;
    }
    return 0;
}
```

Q33) PROBLEM STATEMENT: GIVEN AN INTEGER N, RETURN TRUE IT IS AN ARMSTRONG NUMBER OTHERWISE RETURN FALSE.

AN AMRSTRONG NUMBER IS A NUMBER THAT IS EQUAL TO THE SUM OF

ITS OWN DIGITS EACH RAISED TO THE POWER OF THE NUMBER OF DIGITS.

INPUT

[153, 371, 108]

OUTPUT

[123 , 371]

Code:

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>

using namespace std;

bool isArmStrongNumber(int n, int k) {
    int sum = 0;
    int originalNum = n;
    while (originalNum > 0) {
        int digit = originalNum % 10;
        sum += pow(digit, k);
        originalNum /= 10;
    }
    return sum == n;
}

int main() {
    string input;
    cout << "Enter numbers separated by commas: ";
    getline(cin, input);
    vector<string> arr;
    size_t pos = 0;
    while ((pos = input.find(',')) != string::npos) {
        arr.push_back(input.substr(0, pos));
        input.erase(0, pos + 1);
    }
    arr.push_back(input);
    int flag = 0;
    for (string num_str : arr) {
        int num = stoi(num_str);
        int n = num_str.length();
        if (isArmStrongNumber(num, n)) {
            cout << num << " ";
            flag = 1;
        }
    }
    if (flag == 0) {
        cout << "No Armstrong numbers present";
    }
    return 0;
}
```

Q34) Top candidate need to be printed from the given input.

Input format:

34 Rohit 90 John 85 Bob 92 Alice 110 Aditya

Output:

Aditya: 110

Alice: 100

John: 90

Code:

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

using namespace std;

int main() {
    string s;
    getline(cin, s);
    vector<string> l;
    string token;
    istringstream iss(s);
    while (iss >> token) {
        l.push_back(token);
    }
    int k = stoi(l[0]);
    int N = stoi(l[1]);
    vector<pair<int, string>> arr;

    for (int i = 2; i < l.size(); i += 2) {
        int marks = stoi(l[i]);
        string name = l[i + 1];
        arr.push_back(make_pair(marks, name));
    }

    sort(arr.begin(), arr.end(), greater<pair<int, string>>());

    for (int top = 0; top < k && top < arr.size(); top++) {
        cout << arr[top].second << ": " << arr[top].first << endl;
    }

    return 0;
}
```

Q35) Print all unique combinations.

Input:

[3 0 1 2 3]

5- is the size of the array.

[0,01,012,0123, 1, 12, 123, 2, 23, 3]

Code:

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

int main() {
    std::string input;
    std::getline(std::cin, input);
    std::istringstream iss(input);

    int n;
    iss >> n;

    std::vector<int> arr;
    int temp;
    while (iss >> temp) {
        arr.push_back(temp);
    }

    for (int i = 0; i < n; ++i) {
        for (int j = i; j < n; ++j) {
            for (int q = i; q <= j; ++q) {
                std::cout << arr[q];
                if (q != j)
                    std::cout << " ";
            }
            if (i != n - 1)
                std::cout << ",";
        }
    }

    return 0;
}
```

Q36) Sum of Unique Elements

You are given an integer array nums. The unique elements of an array are the elements that appear exactly once in the array.

Return the sum of all the unique elements of nums.

Example 1:

Input: nums = [1 2 3 4]

Output: 4

Explanation: [1,3], and the sum is 4.

Example 2:

Input: nums = [1 1 1 1 1]

Output: 0

Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
#include <sstream>

using namespace std;

int main() {
    string input;
    getline(cin, input);
    stringstream ss(input);
    vector<int> nums;
    int num;
    while (ss >> num) {
        nums.push_back(num);
    }
    unordered_map<int, int> d;
    for (int i = 0; i < nums.size(); ++i) {
        if (d.find(nums[i]) == d.end()) {
            d[nums[i]] = 1;
        } else {
            d[nums[i]]++;
        }
    }
    int ans = 0;
    for (auto it: d) {
        if (it.second == 1) {
            ans += it.first;
        }
    }
    cout << ans << endl;
    return 0;
}
```

Q37) Given an integer array nums, find the subarray with the largest sum, and return its sum.

Example 1:

nums= [-2 1-3 4-1 2 1-5 4]

Output: 6

Explanation: The subarray [4,-1,2,1] has the largest sum 6.

Example 2:

Input: nums = [5 4 -1 7 8]

Output: 23

Explanation: The subarray [5,4,-1,7,8] has the largest sum 23

Code:

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

using namespace std;

int main() {
    string input;
    getline(cin, input);
    stringstream ss(input);
    vector<int> nums;
    int num;
    while (ss >> num) {
        nums.push_back(num);
    }
    int maxSum = nums[0];
    int currentSum = 0;
    for (int i = 0; i < nums.size(); ++i) {
        currentSum += nums[i];
        if (currentSum > maxSum) {
            maxSum = max(currentSum, maxSum);
        }
        if (currentSum < 0) {
            currentSum = 0;
        }
    }
    cout << maxSum << endl;
    return 0;
}
```

Q38) You are tasked with writing a program to calculate the total shipping cost based on the weight of the package and the distance it needs to travel. The shipping cost is determined by the following criteria:

1. Base money: $5.00

2. Cost per kilogram: $2.00

3. Cost per 10 kilometers: $0.5

Example 1:

10 (w)

100 (D)

output: $30.00

Code:

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

double calculate_total_cost(int weight, int distance) {
    double base_money = 5.00;
    double cost_per_kg = 2.00;
    double cost_per_10_km = 0.50;

    double weight_cost = weight * cost_per_kg;
    double distance_cost = (distance / 10) * cost_per_10_km;

    double total_cost = base_money + weight_cost + distance_cost;
    return total_cost;
}

int main() {
    int weight, distance;

    std::cin >> weight;
    std::cin >> distance;

    double total_cost = calculate_total_cost(weight, distance);

    char output[50];
    sprintf(output, "$%.2f", total_cost);

    std::cout << output << std::endl;

    return 0;
}
```

Q39) Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k.

A subarray is a contiguous non-empty sequence of elements within an array.

Input nums= [-1 2 3-3 1 1 1]

k=3

Output: 6

Code:

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

int subarraySum(vector<int>& arr, int k) {
    unordered_map<int, int> mpp;
    mpp[0] = 1;
    int preSum = 0, cnt = 0;
    for (int i = 0; i < arr.size(); ++i) {
        preSum += arr[i];
        int rest = preSum - k;
        cnt += mpp[rest];
        mpp[preSum] += 1;
    }
    return cnt;
}

int main() {
    string inputLine;
    getline(cin, inputLine);
    istringstream iss(inputLine);
    vector<int> arr;
    int number;
    while (iss >> number) {
        arr.push_back(number);
    }

    int k;
    cin >> k;

    int result = subarraySum(arr, k);
    cout << result << endl;

    return 0;
}
```

Q40) Given a string S(input) consisting of * and #. The length of the string is variable. The task is to find the minimum number of * and # required to make it a valid string. The string is considered valid if the number of * and # are equal. The * and # can be at any position in the string.

Note: The output will be a positive or negative integer based on number of * and # in the input string.
(* > #) : Positive integer
(# > *) : Negative integer
(#=*): 0

Example 1:
Input

###*** → Value of S
Output
0 ---> number of * and # are equal
Example 2:
Input
###***# → Value of S
Output
-1 ---> number of # is more than *
Example 3:
Input
#*** → Value of S
Output
2 ---> number of * is more than #

The input format for testing

The candidate has to write the code to accept one inputs separated by a new line.
First input: Accept a string S without any spaces (consisting of only * and #)

The output format for testing

The output should be a positive integer number (check in e.g.s 1, 2 and 3). Additional messages in output
will cause the failure of test cases.

Constraints
S = {*,#}
0<len(S)<=50

Code Solution:

```
#include<iostream>
 using namespace std;
  int main()
        {
           string str;
           int count = 0, count1 = 0;
           cin >> str;
           for(int i = 0; str[i] != '\0'; i++)
              {
            if(str[i] == '#')
            count++;
            if(str[i] == '*')
            count1++;
    }
        if(count == count1)
        cout << "0";
       else
       cout << count1-count;
}
```

Q41) Given an array Arr[ ] of N integers and a positive integer K. The task is to cyclically rotate the array clockwise by K.
Note : Keep the first of the array unaltered.
Example 1:
5 ---Value of N
{10, 20, 30, 40, 50} ---Element of Arr[]

2 -----Value of K
Output :
40 50 10 20 30
Example 2:
4 ---Value of N
{10, 20, 30, 40} ---Element of Arr[]
1 -----Value of K
Output :
40 10 20 30

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> rotate(int nums[], int n, int k) {
if (k > n)
k = k % n;
vector<int> ans(n);

for (int i = 0; i < k; i++) {
ans[i] = nums[n - k + i];
}
int index = 0;
for (int i = k; i < n; i++) {
ans[i] = nums[index++];
}
return ans;
}
int main()
{
```

```
int Array[] = { 1, 2, 3, 4, 5 };
int N = sizeof(Array) / sizeof(Array[0]);
int K = 2;
vector<int> ans = rotate(Array, N, K);
for (int i = 0; i < N; ++i) {
cout << ans[i] << ' ';
}
}
```

Q42) Given two non-negative integers n1 and n2, where n1<n2. The task is to find the total number of integers in the range [n1, n2](both inclusive) which have no repeated digits.
For example:
Suppose n1=11 and n2=15.
There is the number 11, which has repeated digits, but 12, 13, 14 and 15 have no repeated digits. So, the output is 4.
Example1:
Inout:
11 --- Value of n1
15 -- value of n2
Output:
4
Example 2:
Input:
101 -- value of n1
200 -- value of n2
Output:
72

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
int find(int n1, int n2) {

int count = 0;
for (int i = n1 ; i <= n2 ; i++) {
int num = i;
vector<bool> visited;
visited.assign(10, false);
while (num > 0) {
if (visited[num % 10] == true)
break;
visited[num % 10] = true;
num /= 10;
}
if (num == 0)
count++;
}
return count;
}
int main()
{
int n1 = 101, n2 = 200;
cout << find(n1, n2);
}
```

Q43) Given an array Arr[ ] of N integer numbers. The task is to rewrite the array by putting all multipliers at the end

of the given array.

Note : The order of the numbers which are not the multiplier of 10 should remain unaltered ,and similarly the
order of the numbers which are the multiplier of 10 should remain unaltered.
For example :
Suppose N = 9 and Arr[] = {10, 12, 5, 40, 30, 7, 5, 9, 10}
You have to push all the multiple of 10 at the end of the array Arr[].
Hence the output is : 12 5 7 5 9 10 40 30 10

Solution in C++

```
#include<bits/stdc++.h>
using namespace std;
int main ()
{
int N;
cin>>N;
vector <int> Arr(N);
for(int i=0; i<N; i++)
cin>>Arr[i];
vector<int> A1, A2;
for(int i=0; i<N; i++)
{
if(Arr[i]%10==0)
A2.push_back(Arr[i]);

else A1.push_back(Arr[i]);
}
for(int i=0; i<A1.size(); i++)
Arr[i]=A1[i];
int k=A1.size();
for(int i=0; i<A2.size(); i++)
Arr[k++]=A2[i];
for(int i=0; i<N; i++)
cout<<Arr[i]<<" ";
return 0;
}
```

Q44) Given an array Arr[ ] of N integers and a positive integer K. The task is to Divide the array into two sub array from right after the Kth position and slide the left sub array of K elements to the end.
Example 1:
5 ---Value of N
{10, 20, 30, 40, 50} --- Element of Arr[]
2 -----Value of K
Output :
30 40 50 10 20
Example 2:
4 ---Value of N
{10, 20, 30, 40} ---Element of Arr[]
1 -----Value of K

Output :
20 30 40 10

Solution in C++ :

```
#include<bits/stdc++.h>
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;
queue <int> que;
for(int i=0; i<N; i++)
que.push(Arr[i]);
while(K--)
{
int x= que.front();
que.pop();
que.push(x);
}
while(!que.empty()){
cout<<que.front()<<" ";
que.pop();

}
return 0;
}
```

Q45) For hiring a car, a travel agency charges R1 rupees per hour for the first N hours and then R2 rupees per hour.
Given the total time of travel in minutes in X.
The task is to find the total travelling cost in rupees.
Note : While converting minutes into hours, ceiling value should be considered as the total number of hours.
For example : If the total travelling time is 90 minutes, i.e. 1.5 hours, it must be considered as 2 hours.
Example :
Input :
20 -- r1
4 -- n
40 -- r2
300 -- x
Output :
120
Explanation :

Total travelling hours = 300 / 60 = 5 hours
Rupees 20 / hours for first 4 hours = 20 * 4 = 80 rupees
Rupees 40 / hours in 5th hour = 40 * 1 = 40 rupees
Hence, the total travelling cost = 80 + 40 = 120 rupees

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
int main () {
int r1;
cin >> r1;
int n;
cin >> n;
int r2;
cin >> r2;
int x;
cin >> x;
int total = 0;
int hours = ceil(x * 1.0 / 60);
if (n > hours) {
total += (hours * r1);
} else {
total += n * r1;
hours -= n;
total += hours * r2;
}
cout << total;
return 0;
}
```

Q46) There is a bag with three types of gemstones: Ruby of type R, Garnet of G, and Topaz of type T.

Write a program to find the total number of possible arrangements to make a series of gemstones where no two gemstones of the same type are adjacent to each other.
Example 1:
Input:
1-- count of Ruby
1-- count of Garnet
0-- count of Topaz

Output:
2
Arrangements are RG and GR.
Example 1:
Input:
1-- count of Ruby
1-- count of Garnet
1-- count of Topaz
Output:
6

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
#define mod 1000000007
int dp[21][21][21][4];
int countPossibleWays(int a, int b, int c, int prev) {
if (dp[a][b][c][prev] != -1) {
return dp[a][b][c][prev];
}
else if (a == 0 && b == 0 && c == 0) {
dp[a][b][c][prev] = 1;
return dp[a][b][c][prev];
}

long long ans = 0;
if (prev == 0) {
```

```
if (b != 0)
ans = (ans + countPossibleWays(a, b - 1, c, 1)) % mod;
if (c != 0)
ans = (ans + countPossibleWays(a, b, c - 1, 2)) % mod;
}
else if (prev == 1) {
if (a != 0)
ans = (ans + countPossibleWays(a - 1, b, c, 0)) % mod;
if (c != 0)
ans = (ans + countPossibleWays(a, b, c - 1, 2)) % mod;
}
else {
if (a != 0)
ans = (ans + countPossibleWays(a - 1, b, c, 0)) % mod;
if (b != 0)
ans = (ans + countPossibleWays(a, b - 1, c, 1)) % mod;
}
dp[a][b][c][prev] = ans;
return ans;
}
void solve(int a, int b, int c) {
memset(dp, -1, sizeof(dp));
long long ans = 0;
if (a != 0)
ans = (ans + countPossibleWays(a - 1, b, c, 0)) % mod;
if (b != 0)
ans = (ans + countPossibleWays(a, b - 1, c, 1)) % mod;
if (c != 0)
ans = (ans + countPossibleWays(a, b, c - 1, 2)) % mod;
cout << ans;
}
int main () {
```

```
int a, b, c;
cin >> a >> b >> c;
solve(a, b, c);

return 0;
}
```

Q47) Given a sentence cstr, written in a camel case (i.e. every word starts with an uppercase letter and there is no space or punctuation between two consecutive words). The task is to put one space after every word and convert every uppercase letter to lowercase.
Example 1:
Input :
ThisIsAnAutomationEra
Output:
this is an Automation Era--- Value of cstr
Output:
this is an automation era
Example 2:
Input:
HeyYou--- Value of cstr
Output:
hey you
Constraints:
● The string cannot contain space.
● Size of cstr <= 500
Input format for testing:
● The candidate has to write the code to accept a single string cstr consisting of only letters of the alphabet with no space.

Output format for testing:
● The output should be a string only. (See the output format in Example 1 and Example 2)
● Additional messages in the output will result in the failure of your test cases.
Instructions:
● The System does not allow any kind of hard-coded input value/values.
● The written program code by the candidate will be verified against the input which is supplied from the System.

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
string cstr;
cin>>cstr;
string res="";
for(int i=0; i<cstr.size(); i++){
if( 'A'<=cstr[i] and cstr[i]<='Z')
{
if(i!=0)
res += " ";
res += (cstr[i] + 32);
}
else res += cstr[i];
}
cout<<res;
return 0;
}
```

Q48)A company is organizing a fun game for its employees. N number of employees are participating in this game.
Each employee can either compete in the game as an individual or as a pair with another employee. The task is
to find the number of different ways in which N number of employees can be single or can be paired up.
Condition:
● If the total count of employees is odd, then pairing is not allowed.
● If the total count of employees is even, then employees can be single or can be paired up.

Instructions:
● Each employee can be paired only once.
● Each employee can only compete once
Example 1:
Input:
3---- Number of employees
Output:
1
Explanation:
N= 3 i.e, Odd Number
[1],[2],[3]: all employees are single.
Example 2:
Input:
4--- Numbers of employees
Output:

10
Explanation:
N= 4 i.e, Even Number
[1],[2],[3],[4]: all employees are single.
[1],[2,3],[4]: 3 and 4 are paired but 1 and 2 are single
[1,2],[3],[4]: 2 and 3 are paired but 1 and 4 are single
[1,3],[2],[4]: 1 and 3 are paired but 2 and 4 are single
[1,4],[2],[3]: 1 and 4 are paired but 2 and 3 are single
[2,4],[3],[1]: 2 and 4 are paired but 3 and 1 are single
[1,2], [3,4] : pairs of 1 and 2, 3 and 4
[1,3], [2,4] : pairs of 1 and 3, 2 and 4
[1,4], [2,3] : pairs of 1 and 4, 2 and 3

Note: [2,3] and [3,4] are considered the same
Constraints:
● 1< N<= 10
Input Format for testing:
● The Candidate has to write the code to accept a positive integer number representing count of employees
Output Format for testing:
● The output should be a positive integer only. (See the output format in example).
● Additional messages in the output will result in the failure of test cases.
Instructions:
● The System does not allow any kind of hard-coded input value/values.
● The written program code by the candidate will be verified against the input which are supplied from the system.

Solution in C++:

```
#include<bits/stdc++.h>
using namespace std;
int count_ways( int N){
if(N%2!=0)
return N;
int t[N+1];
for(int i=0; i<=N; i++)
{
if(i<=2)
t[i]=i;
else t[i] = t[i-1] + (i-1)*t[i-2];
}

return t[N];
}
int main(){
int N;
cin>>N;
int res = count_ways(N);
cout<<res;
return 0;
}
```

Q49) Write a program to solve the Towers of Hanoi problem with 'n'' disks.

Code Solution:

```
 public class TowersOfHanoi:
 {
  public static void main(String[] args) {
 int n=3;
 towerOfHanoi(n, 'A", 'C", 'B');
}
static void towerOfHanoi(int n, char from rod, char to_rod, char
aux_rod) {
if(n==1) {
System.out.println("Move disk 1 from rod " + from rod + " to rod " +
to_rod);
return;
}
towerOfHanoi(n - 1, from rod, aux_rod, to_rod);
System.out.println("Move disk " +n +" from rod " + from_rod + " to rod "
+ to_rod);
towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
}
```

Q50) Write a program to find the Longest Common Subsequence (LCS) in two strings.

Code Solution:

```
 public class LCS {
 public static void main(String[] args) {
String s1 = "AGGTAB";
String s2 = "GXTXAYB";
int result = les(s1, s2, s1.length(), s2.length());
System.out.println("Length of LCS is " + result);
}
static int lcs(String s1, String s2, int m, int n) {
if(m==0||n==0)
return 0;
if (s1.charAt(m - 1) == s2.charAt(n - 1))
return 1 + lcs(sl, s2, m- 1,n-1);
else
return Math.max(lcs(s1, s2, m, n - 1), lcs(s1, s2, m - 1, n));
}
}
```