BIZOTIC
For the love of education

COMPANY SPECIFIC SERIES
ACCENTURE – CODING - TRAINER HANDOUT

PROGRAMS ASKED IN RECENT ACCENTURE DRIVES:

1. Edward's Birthday

It is Edward's birthday today. His friends have bought him a huge circular cake.
Edward wants to find out the maximum number of pieces he can get by making exactly N straight vertical cuts on the cake.
Your task is to write a function that returns the maximum number of pieces that can be obtained by making N number of cuts.
Note: Since the answer can be quite large, modulo it by 1000000007
Input Specification:
input1: An integer N denoting the number of cuts
Output Specification:
Return the maximum number of pieces that can be obtained by making N cuts on the cake

Example: Input	Output	Explanation
1	2	Given the above scenario, if we make 1 cut on the cake, the maximum number of pieces of cake we obtain is 2 and 2%1000000007=2. Hence, 2 is returned as output.
5	16	Given the above scenario, if we make 5 cuts on the cake, the maximum number of pieces of cake we obtain is 16 and 16% 1000000007=16. Hence, 16 is returned as output.

Code Solution in Python

```
class Solution:
def maximumCuts(self, n):
M = 1000000007
num=((n*(n+1))//2) +1
return num % M
if __name__ == '__main__':
n = int(input())
ob = Solution()
ans = ob.maximumCuts(n)
print(ans)
```

Code Solution in JAVA

```
import java.util.*;
import java.lang.*;
import java.io.*;
class Main
{
public static void main (String[] args) throws IOException
{
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
Solution ob = new Solution();
System.out.println(ob.maximumCuts(n));
}
}
class Solution
{
static long maximumCuts(int n)
{
long M = 1000000007;
if(n==1 )
{
return 2;
}
if(n==0)
{
return 1;
}
return (n + maximumCuts(n-1))%M;
}
}
```

Code Solution in C++

```
#include<bits/stdc++.h>
using namespace std;
int maximumCuts(int n){
const unsigned int M = 1000000007;
int sum=1;
while(n!=0)
sum+=n--;
return sum%M;
}
int main(){
int n;
cin >> n;
int ans = maximumCuts(n);
cout << ans <<"\n";
return 0;
}
```

Code Solution in C

```
#include<stdio.h>
int maximumCuts(int n){
const unsigned int M =
1000000007;
int sum=1;
while(n!=0)
sum+=n--;
return sum%M;
}
int main(){
int n;
scanf("%d",&n);
int ans = maximumCuts(n);
printf("%d\n",ans);
return 0;
}
```

2. Regions on a Plane

Problem statement
Mr. Professor is a great scientist, but he is not able to find a solution to one problem. There are N straight lines that are not parallel, and no three lines go through the same point. The lines divide the plane into M regions. Write a function to find out the maximum number of such regions he can get on the plane.
Input Specification:
input1: An integer N representing the number of straight lines (0 <=N<= 100)
Output Specification:
Return the maximum number of regions

Example:

Input	Output	Explanation
2	4	Given the above scenario, 2 lines divide the plane into 4 regions. Therefore, 4 is returned as the output.
3	7	Given the above scenario, 3 lines divide the plane into 7 regions. Therefore, 7 is returned as the output

Code Solution in Python

```
class Solution:
def maxRegions(self, n):
num = n * (n + 1) // 2 + 1
return num
if __name__ == '__main__':
n = int(input())
ob = Solution()
ans = ob.maxRegions(n)
print(ans)
```

Code Solution in JAVA

```
import java.util.*;
import java.lang.*;
import java.io.*;
class Main
{
public static void main (String[] args) throws IOException
{
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
Solution ob = new Solution();
ob.maxRegions(n);
}
}
class Solution
{
```

```
public static void maxRegions(int n)
{
int num;
num = n * (n + 1) / 2 + 1;
System.out.println(num);
}
}
```

Code Solution in C++

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

Code Solution in C

```
#include<stdio.h>
void maxRegions(int
n)
{
int num;
num = n * (n + 1) / 2
+ 1;
printf("%d",num);
}
int main()
{
int n;
scanf("%d",&n);
maxRegions(n);
return 0;
}
```

3. Sum of odd integers in array

Problem statement

An odd number is an integer which is not a multiple of 2.

You are required to implement the following function:

Int SumOddIntegers(int arr[], int n);

The function accepts an integer array ‘arr’ of length ‘n’ as its argument. You are required to calculate the sum of all odd integers in an array ‘an’ and return the same

Note:

Array can have negative integers

n > 0

Computed values lie within integer range

Example:

Input:

arr: 1 4 6 7 10 12 11 5

n: 8

Output:

24

Explanation:

The odd integers in array {1, 4, 6, 7, 10, 12, 11, 5} are {1, 7,11, 5} and their sum is 24, hence 24 is returned.

The custom input format for the above case:

8

1 4 6 7 10 12 11 5

(The first line represents ‘n’ the second line represents the elements of the array)

Sample Input

arr: 2 4 9 7 11 13 25 31 6 8 10 24

n: 12

Sample Output

96

The custom input format for the above case:

12

2 4 9 7 11 13 25 31 6 8 10 24

(The first line represents ‘n’ the second line represents the elements of the array)

Code Solution in C:

```c
#include <stdio.h>
int sumAllOdds(int arr[], int size) {
  int sum = 0;
  for(int i = 0 ; i < size; i++) {
    if(arr[i] % 2 != 0) {   // checks if number is odd
      sum += arr[i];
    }
  }
  return sum;
}

int main() {
  int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
       }
    printf("%d\n",sumAllOdds(a,n));
  return 0;
}
```

Code Solution in C++ :

```cpp
#include <bits/stdc++.h>
using namespace std;
int sumAllOdds(int arr[], int size) {
  int sum = 0;
  for(int i = 0 ; i < size; i++) {
    if(arr[i] % 2 != 0) {   // checks if number is odd
      sum += arr[i];
    }
  }
  return sum;
}

int main() {
   int n;
    cin>>n;
    int a[n];
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
     }
   cout<<sumAllOdds(a,n)<<endl;
  return 0;
}
```

Code Solution in Java Code:

```java
import java.util.*;
import java.lang.*;
import java.io.*;

class sumAllOdds{
  public static void main(String args[])
  {
    int odd = 0;
    Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i <n; i++){
            arr[i] = in.nextInt();
            }
    for (int i = 0; i <n; i++) {
      if ((arr[i] % 2) != 0)
        odd += arr[i];
    }
    System.out.println(odd);
  }
}
```

4. Inversion count in array

Problem statement

Let j and k be two indices in an array A.
If j<K and A[j] > A[k], then the pair (j,k) is known as an "Inversion pair".
You are required to implement the following function:
int InversionCount(int *A, int n);
The function accepts an array 'A' of 'n' unique integers as its argument. You are required to
Calculate the number of 'Inversion pair' in an array A, and return.

Note:

If 'A' is NULL (None in case of python). return – 1
I_f_ _'n_' _<_2_,_ _r_e_t_u_r_n_ _0_ _

Example:

Input:

A: 1 20 6 4 5
n: 5

Output:

5

Explanation:

The inversion pair in array A are (20,6),(20,4),(20,5),(6,4) and (6,5), the count of the inversions are 5, hence 5 is returned.
The custom input format for the above case:

5
1 20 6 4 5
(The first line represents the size of the array, the second line represents the elements of the array)

Sample Input

A: 13 10 9 6 21 15 14
n: 7

Sample Output

9

The custom input format for the above case:

7
13 10 9 6 21 15 14

Code Solution in C:

```c
#include <stdio.h>

int getInvCount(int arr[], int n)
{
  int inv_count = 0;
  for (int i = 0; i < n - 1; i++)
    for (int j = i + 1; j < n; j++)
      if (arr[i] > arr[j])
        inv_count++;

  return inv_count;
}

int main() {
  int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
      }
    printf("%d\n",getInvCount(a,n));
  return 0;
}
```

Code Solution in C++:

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

int getInvCount(int arr[], int n)
{
  int inv_count = 0;
  for (int i = 0; i < n - 1; i++)
    for (int j = i + 1; j < n; j++)
      if (arr[i] > arr[j])
        inv_count++;

  return inv_count;
}

int main() {
  int n;
    cin>>n;
    int a[n];
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
   cout<< getInvCount(a, n)<<endl;
  return 0;
}
```

Code Solution in Java:

```java
import java.util.*;
import java.lang.*;
import java.io.*;

class InvCount {
  public static void main(String[] args)
  {
     Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i <n; i++){
            arr[i] = in.nextInt();
            }
         int inv_count = 0;
    for (int i = 0; i < n - 1; i++)
      for (int j = i + 1; j < n; j++)
        if (arr[i] > arr[j])
          inv_count++;
  System.out.println(inv_count);
  }
}
```

PROGRAMS FROM LAST YEAR'S DRIVE

Program 1 - Balance fruits (Category: Medium)

Implement the following function:
int BalanceFruits(int a, int m, int rs);
You have a basket full of apples and mangoes, your job is to make the number of apples and given a function that accepts three integers 'a', 'm' and 'rs' as its argument where 'a' and a basket respectively and 'rs' is the rupees that you have. Implement the function to balance the basket.

- If 'a' > 'm', then buy (a - m) mangoes at the rate of Rs 1 per mango.
- If 'a' < 'm', then sell (m - a) mangoes at the rate of Rs 1 per mango.

Return the total rupees left with you after balancing the fruits.
Assumption:

- a > = 0, m > = 0 and rs > = 0
- rs > = (a - m)

Note: If a = m, return rs unchanged

Example
Sample Input
a : 8
m : 4
Rs : 6
Sample Output
2

Explanation: Since a > m, (a - m) mangoes are bought at Rs 1 per mango, so rs = 6 - 4 = 2. Thus, output is 2.
Case 1
Input
8 4 6
Output
2

Case 2
Input
7 4 6
Output
3

Program

```
#include<iostream>
```

```
using namespace std;
int main()
{
	int a, m, rs;
	cin >> a >> m >> rs;
	if(a > m)
	{
	cout << rs - (a - m);
	}
	else if( a == m)
	{
	cout << rs;
	}
	else if( a < m)
	{
	cout << rs + (m - a);
	}
}
```

Program 2 - Tallest Tree (Category: Medium)

A TreeHeight is represented by the following structure:

```
struct TreeHeight
{
 int feets;
 int inches;
};
```

You are given a function,

int TallestTree(struct TreeHeight trees[], int n);

The function accepts an array 'trees' of type 'TreeHeight' consisting of 'n' elements as its argument. 'TreeHeight' consists of two integers 'feets' and 'inches' which represents height of a tree. Implement the function to find the tallest tree among the 'trees' and return its height (in inches).

Height of a tree = (12 * feets) + inches

Assumption:

- feets > 0
- 0 > = inches > = 11

Note:

- Computed value lies within integer range.
- Return -1 if trees is null (None, in case of Python).
- trees(0) is the first element.

Example:

Sample Input:

Feets Inches
10 4
23 5
21 2
27 7

Sample Output:

331

Explanation:

Height of trees (in inches):

- trees[0] = (10 * 12) + 4 = 124
- trees[1] = (23 * 12) + 5 = 281
- trees[2] = (21 * 12) + 2 = 254
- trees[3] = (27 * 12) + 7 = 331

Maximum of {124, 281, 254, 331} = 331, thus output is 331.

Case 1

Input

4
10 4
23 5
21 2
27 7

Output

331

Case 2

2
46 10
55 6

Output

666

Program

```
#include<iostream>
using namespace std;
struct TreeHeight
{
        int feets;
        int inches;
};
 int main()
{
        int n;
        cin >> n;
        struct TreeHeight arr[n];
        for(int i = 0; i < n; i++)
        {
        cin >> arr[i].feets;
        cin >> arr[i].inches;
        }
        int max = 0;
        for(int i = 0; i < n; i++)
        {
        int temp = arr[i].feets * 12 + arr[i].inches;
        if(max < temp)
        max = temp;
        }
        cout << max;
}
```

Program 3 - Fragments divisible by eleven (Category: Medium)

You are given a function,
def DivisibilityByEleven(num):
The function accepts an integer 'num' as input. You have to implement the function such that it returns the number of contiguous integer fragments of 'num' that are divisible by 11. Contiguous integer fragments of a number, say 1273, are:
Example:
Input:
1215598
Output:
4

Explanation: 55, 121, 12155 and 15598 are contiguous fragments of the number 1215598 that are divisible by 11.

Case 1:
Input
1215598
Output
5

Case 2:
Input
55
Output
1
Program

```
#include<bits/stdc++.h>
using namespace std;
 int main()
{
        int n;
        cin >> n;
                int temp = n;
        int sum, count = 0;
        vector <int> v;
        int len = 0;
        while(n != 0)
        {
    v.push_back(n % 10);
        n = n / 10;
```

```
        }
reverse(v.begin(), v.end());
        for(int i = 0; i < v.size(); i++)
        {
        sum = v[i];
        for(int j = i+1; j < v.size(); j++)
        {
        sum = sum * 10 + v[j];
        if(sum % 11 == 0)
           count++;
        }  }
        cout << count; }
```

PROGRAM 4 - Convert to Palindrome (Category: Medium)

A palindrome is a sequence of character that has the property of reading the same way either in direction. You are given a function,

char *ConvertPalindrome(char* str)

The function accepts the string str. Implement the function to find and return the minimum characters required to append at the end of string str to make it palindrome

Assumption

1.String will contain only lower case English alphabets.

2.Length of string is greater than or equal to 1

Note:

1. If string is already palindrome return “NULL”.
2. You have to find the minimum characters required to append at the end of string to make it palindrome.

Example:

Sample Input

abcdc

Sample Output

ba

Explanation: If we append ‘ba’ at the end of the string ‘abcdc’ it becomes abcdcba i.e. a palindrome.

Case 1:

Input

abcdc

Output

ba

Case 2:

Input

ab

Output

a

Program

```
#include<bits/stdc++.h>
using namespace std;
bool checkPalindrome(string s,int i,int j)
{
   while(i<j)
        {
        if(s[i] != s[j])
        return false;
```

```
        i++;
        j--;
        }
        return true;
}
int main()
{
        string inp;
   cin>>inp;
        int n=inp.length();
        int i;
   for(i=0;i<n;i++)
        {
      if(checkPalindrome(inp,i,n-1))
        break;
        }
        for(int j = i-1; j >= 0; j--)
        cout << inp[j];
        return 0;
}
```

Program 5 - Count Specific Numbers (Category: Difficult)

You are required to implement the following function:
int CountSpecificNumbers(int m, int n);
The function accept two arguments m and n which are integers. You are required to calculate the count of numbers having only 1, 4 and 9 as their digits between the numbers lying in the range m and n both inclusive, and return the same. Return -1 if m>n.

Example:

Sample Input:

100
200

Sample Output:

9

Explanation: The numbers between 100 and 200, both inclusive having only 1, 4 and 9 as their digits are 111, 114, 119, 141, 144, 149, 191, 194, 199. The count is 9 hence 9 is returned.

Case 1:

Input

100
200

Output

9

Case 2:

Input

1
100

Output

12

Program

```
#include<bits/stdc++.h>
using namespace std;
int CountSpecificNumber(int m,int n);
int main()
{
        int m,n;
        cin>>m>>n;
        cout<<CountSpecificNumber(m,n);
        return 0;
}
int CountSpecificNumber(int m,int n)
```

```
{
	if(m<=n)
	{
		int i,count=0;
		for(i=m;i<=n;i++)
		{
			int num=i,flag=1;
			while(num)
			{
				int n = num%10;
				num=num/10;
				if(n == 1 || n == 4 || n == 9)
					continue;
				else
				{
					flag=0;
					break;
				}
			}
			if(flag==1)
				count++;
		}
	return count;
	}
	return -1;
}
```

Program 6 - Picking Tickets (Category: Difficult)

Consider an array of n tickets prices, tickets. A number, m, is defined as the size of some subsequence, s, of tickets where each element covers an unbroken range of integers. That is to say if you were to sort the elements in s, the absolute difference between any elements *j* and *j+1* would be either 0 or 1. Determine the maximum length of a subsequence chosen from the ticket array.

Example:

Tickets = [8,5,4,8,4]

Valid subsequences, sorted, are{4,4,5} and {8,8}.These subsequences have m, values of 3 and 2, respectively. Return 3.

Return:

Int: an integer that denotes the maximum possible value of m.

Constraints:

1<= n <=105

1<= tickets[i] <= 109

Example:

Sample Input:

STDIN FUNCTION

--- ---

4 -> tickets [] size n=4

4 -> tickets = [4,13,2,3]

13

2

3

Sample Output:

3

Explanation: There are two subsequences of tickets that contain consecutive integers when sorted: {2,3,4} and {13}. These subsequences have m values of 3 and 1, respectively. Return the maximum values of m, which is 3.

Case 1:

Input

4

4

13

2

3

Output

3

Case 2:
Input
5
8 5 4 8 4
Output
3

Program

```
#include<bits/stdc++.h>
using namespace std;
int maxSubsequence(int a[],int n)
{
	int max=0,i,count=0,flag=0;
	for(i=0;i<n-1;i++)
	{
		int dif = a[i+1]-a[i];
		if(dif==1||dif==0)
		{
			count++;
			flag=1;
		}
		else
		{
			if(count>max)
				max=count;
			count=0;
		}
	}
	if(flag)
		return count>max?count+1:max+1;
	else
		return count;
}
int main()
{
	int n;
	cin>>n;
	int a[n];
	for(int i=0;i<n;i++)
		cin>>a[i];
```

```
    sort(a,a+n);
    cout<<maxSubsequence(a,n);
}
```

Program 7 - Count Occurrences of Digit (Category: Medium)

You are required to implement the following function:
int CountDigiOoccurrences(int l, int u, int x);
The function accepts 3 positive integers ‘l’, 'u' and 'x' as its arguments. You are required to calculate the number of occurrences of a digit 'x' in the digits of numbers lying in the range between ‘I’ and 'u' both inclusive, and return the same.

Note:

- I < u
- 0 < x < 9

Example:
Sample Input:
I: 2
u: 13
x: 3
Sample Output:
2
Explanation: The number of occurrences of digit 3 in the digits of numbers lying in the range [2, 13] both inclusive is 2, ie (3, 13), hence 2 is returned.

Case 1:
Input
2 13 3
Output
2
Case 2:
Input
1 30 2
Output
13

Program

```
#include<iostream>
using namespace std;
int countDigitOccurrences(int l, int u, int x)
```

```
{
        int rem, count = 0;
        for(int i = l; i <= u; i++)
        {
        int temp = i;
      while(temp != 0 )
        {
        if(temp % 10 == x)
           count++;
        temp = temp / 10;
        }
        }
        return count;
}
int main()
{
  int l, u, x;
  cin >> l >> u >> x;
  cout << countDigitOccurrences(l, u, x);
  return 0;
}
```

Program 8 - Find the BMI (Category: Easy)

A person's body mass index is a simple calculation based on height and weight that classifies the person as underweight, overweight, or normal. The formula for metric unit is,
BMI = weight in kilograms / (height in mesters)2
You are given a function,
Int GetBMICategory(int weight, float height);
The function accepts the weight (an integer number) and height (a floating point number) of a person as its arguments. Implement the function such that it calculations the BMI of the person and returns an integer, the person's BMI category as per the given rules:
1. If BMI < 18, return 0.
2. If 18 >= BMI < 25, return 1.
3. If 25 >= BMI <30, return 2.
4. If 30 >= BMI < 40, return 3.
5. If BMI >= 40, return 4.

Note:
Weight > 0 and its unit is kilogram.
Height > 0 and its unit is metre.
Compute BMI as a floating-point.

Example:
Input:
42
1.54
Output:
0

Explanation: The person's weight is 42Kg and height is 1.54 metres BMI = weight in kilograms / (height in meters)2 = 42/(1.54 * 1.54) = 17.70
Since, BMI < 18, as per given rules, the output is 0.

Sample Input:
62
1.52
Sample Output:
2

Case 1:
Input
42
1.54

Output
0

Case 2:
Input
62
1.52
Output
2

Program
```
#include <iostream>
using namespace std;
int findBMI(int weight,float height){
   float ans = (float) (weight / (height * height));
        if(ans < 18)
        return 0;
        else if(ans >= 18 && ans < 25)
        return 1;
        else if(ans >= 25 && ans < 30)
        return 2;
        else if(ans >= 30 && ans < 40)
        return 3;
        else
        return 4;
}

int main() {
        int weight;
        float height;
        cin>>weight>>height;
        cout<<findBMI(weight,height);

        return 0;
}
```

Program 9 - Numerological Reduction (Category: Easy)

Numerological reduction of a number is the sum of digits calculated recursively until the resulting value is a single digit. You are given a function, int FindNumerologicalReduction(int n);

The function accepts a positive integer ‘n’. Implement the function to find and return the numerological reduction of ‘n’.

Assumption: n> = 0

Example:

Sample Input:

3245

Sample Output:

5

Explanation:

3245 = 3+2+4+5 = 14

14 = 1+4 = 5

Hence 5 is the numerological reduction of 3245.

Case 1:

Input

3245

Output

5

Case 2:

Input

12345

Output

6

Program

```
#include <iostream>
using namespace std;
int main()
{
  int n;
 cin >> n;
 if(n % 9 == 0)
        cout << "9";
 else
        cout << n % 9;

 return 0;
```

}

Program 10 - Email ID Obfuscation (Category: Difficult)

Personal information like email id and other contact details need to be handled in a way so that privacy of the user is safeguarded. One way to do this is to hide or partially hide such information, also called obfuscation of the information.

Given the email id as input, the program should obfuscate the id as follows:

1) For the given mail id, the part that comes before @ is referred to as the first part. If there is no first @ character, the mail id is invalid.

2) If the first part of the email id is less than or equal to 5 characters in length, replace all characters in the first part with *

3) If the first of the email id is greater than 5 characters in length, print the first 3 characters as it is and replace the remaining characters of the first part with *

4) If the email id is invalid, print Invalid Input.

Example 1:

Input :

abc@gmail.com

Output :

***@gmail.com

Explanation: Here the first part is 'abc'. This is less than 5 characters in length, so replace all characters in the first part with *.

Example 2:

Input :

abcdefghi@gmail.com

Output :

abc******@gmail.com

Explanation: Here the first part is 'abcdefghi'. This is more than 5 characters in length, so replace all characters in the first part except the first with *.

Complete the function **obfuscateMailId** in the editor. The function must print the obfuscate mail id as per the rules mentioned. If the mail id is invalid, it should print Invalid Input. **obfuscateMailId** has the following parameter(s) :

mailid : a string representing the mail id

Constraints :

1) A valid mail id should contain the character @ (e.g. myid_at_gmail is invalid)

2) A valid mail id should have first part with at least one or more characters (e.g. @myid is invalid)

Category: Easy

Program 1: Sum of numbers divisible by both 3 and 5

You are required to implement the following function:
Int Calculate (int m, int n);
The function accepts 2 positive integer ‘m’ and ‘n’ as its arguments.You are required to calculate the sum of numbers divisible both by 3 and 5, between ‘m’ and ‘n’ both inclusive and return the same.

Note: 0 < m <= n

Example
Input:
m : 12
n : 50
Output
90

Explanation: The numbers divisible by both 3 and 5, between 12 and 50 both inclusive are {15, 30, 45} and their sum is 90.

Sample Input
m : 100
n : 160
Sample Output
405

Case 1
Input (stdin)
12
50
Output (stdout)
90

Case 2
Input (stdin)
20
30
Output (stdout)
30

Program
#include<iostream>

```
using namespace std;

int Calculate(int, int);
int main()
{
   int m, n, result;
   cin>>m;
   cin>>n;
   result = Calculate(n,m);
   cout<<result;
   return 0;
}

int Calculate(int n, int m)
{
   int i, sum = 0;
   for(i=m;i<=n;i++)
   {
      if((i%3==0)&&(i%5==0))
      {
         sum = sum + i;
      }
   }
   return sum;
}
```

Program 2: Table of Numbers

Write a program to display the table of a number and print the sum of all the multiples in it.

Case 1
Input (stdin)
12
Output (stdout)
12 24 36 48 60 72 84 96 108 120
660

Case 2
Input (stdin)
5
Output (stdout)
5 10 15 20 25 30 35 40 45 50
275

Program

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

int main()
{
   int n, i, value=0, sum=0;
   cin>>n;
   for(i=1; i<=10; ++i)
   {
      value = n * i;
      cout<<value<<" ";
      sum=sum+value;
   }
   cout<<endl<<sum ;
return 0;
}
```

Program 3

Instructions: You are required to write the code. You can click on compile and run anytime to check compilation/execution status. The code should be logically/syntactically correct.

Question: Write a program in C such that it takes a lower limit and upper limit as inputs and print all the intermediate pallindrome numbers.

TestCase 1:

Input :

10 , 80

Expected Result:

11 , 22 , 33 , 44 , 55 , 66 , 77.

Test Case 2:

Input:

100,200

Expected Result:

101 , 111 , 121 , 131 , 141 , 151 , 161 , 171 , 181 , 191.

Program:

```
 #include<stdio.h>
int main()
{
int i, n, reverse, d,f,l;
printf("enter the starting \n",f);
scanf("%d",&f);
printf("enter the ending\n",l);
scanf("%d",&l);
for (i = f; i <= l; i++)
{
reverse = 0;
n = num;
while (n != 0)
{
d = n % 10;
reverse = reverse * 10 + d;
n = n / 10;
}
if (i == reverse)
printf("%d ",i);
}
```

```
return 0;
}
```

Program 4

Instructions: You are required to write the code. You can click on compile & run anytime to check the compilation/ execution status of the program. The submitted code should be logically/syntactically correct and pass all the test cases.

Ques: The program is supposed to calculate the distance between three points.

For

x1 = 1 y1 = 1

x2 = 2 y2 = 4

x3 = 3 y3 = 6

Distance is calculated as : $\mathbf{sqrt(x_2-x_1)^2 + (y_2-y_1)^2}$

Program:

```
#include <stdio.h>
#include <math.h>

int isDistance(float *pt1, float *pt2, float *pt3)
{
float a, b, c;
a = sqrt(((pt2[0]-pt1[0])*(pt2[0]-pt1[0]))+((pt2[1]-pt1[1])*(pt2[1]-pt1[1])));
printf("%f",a);
b = sqrt(((pt3[0]-pt2[0])*(pt3[0]-pt2[0]))+((pt3[1]-pt2[1])*(pt3[1]-pt2[1])));
printf("%f",b);
c = sqrt(((pt3[0]-pt1[0])*(pt3[0]-pt1[0]))+((pt3[1]-pt1[1])*(pt3[1]-pt1[1])));
printf("%f",c);
}

int main()
{
int t;
float p1[2], p2[2], p3[2];
printf("enter x1 and y1 : ");
scanf("%f%f",&p1[0],&p1[1]);
printf("enter x2 and y2 : ");
scanf("%f%f",&p2[0],&p2[1]);
printf("enter x3 and y3 : ");
scanf("%f%f",&p3[0],&p3[1]);
t = isDistance(&p1, &p2, &p3);
```

```
printf("%d",t);
return 0;
}
```

Program 5: Operation Choices

You are required to implement the following function.

int OperationChoices(int c, int a, int b)

The function accepts 3 positive integers ‘a’ , ‘b’ and ‘c ‘ as its arguments. Implement the function to return.
(a+ b) , if c=1
(a - b) , if c=2
(a * b) , if c=3
(a / b) , if c =4
Assumption: All operations will result in integer output.

Example:
Input
c:1
a:12
b:16
Output:
Since ‘c’=1, (12+16) is performed which is equal to 28, hence 28 is returned.

Sample Input
c: 2
a: 16
b: 20
Sample Output
-4

Case 1
Input (stdin)
2
16
20
Output (stdout)
-4

Case 2
Input (stdin)
1
12
16
Output (stdout)
28

Category: Medium

Program 1

Robert is expert in strings where he challenges everyone to write a program for the below implementation.
Two strings and comprising of lower case English letters are compatible if they are equal or can be made equal by following this step any number of times:
Select a prefix from the string (possibly empty), and increase the alphabetical value of all the characters in the prefix by the same valid amount.
For example if the string is abc and we select the prefix ab then we can convert it to bcc by increasing the alphabetical value by 1. But if we select the prefix abc then we cannot increase the alphabetical value.
Your task is to determine if given strings and are compatible.

Input format
First line: String A
Next line: String B

Output format:
For each test case, print YES if string can be converted to string , otherwise print NO.

Constrain: 1<=(len of A,B)<1000005

Sample Input:
abaca
cdbda
Sample Output:
YES

Explanation: The string abaca can be converted to bcbda in one move and to cdbda in the next move.

Test Case 1
Input
abaca cdbda

Output
YES
Test Case 2
Input
abcda abcda
Output
NO

Program:

```
#include<iostream>
#include<string.h>
using namespace std;
int main()
{
   char str1[1000005];
   char str2[1000005];
   cin>>str1>>str2;
   int max = str2[0] - str1[0];
   if(max > 0)
   {
      for(int i =1;i<strlen(str1);i++)
      {
         int x = str2[i] - str1[i];
         if(x > max)
         {
            cout<<"NO";
            return 0;
         }
      }
      cout<<"YES";
   }
   else
      cout<<"NO";
   return 0;
}
```

Program 2: Carry digit

Problem Statement

A carry is a digit that is transferred to left if sum of digits exceeds 9 while adding two numbers from right-to-left one digit at a time

You are required to implement the following function.

Int NumberOfCarries(int num1 , int num2);

The functions accepts two numbers 'num1' and 'num2' as its arguments. You are required to calculate and return the total number of carries generated while adding digits of two numbers 'num1' and ' num2'.

Assumption: num1, num2>=0

Example:

Input

Num 1: 451

Num 2: 349

Output

2

Explanation:

Adding 'num 1' and 'num 2' right-to-left results in 2 carries since (1+9) is 10. 1 is carried and (5+4=1) is 10, again 1 is carried. Hence 2 is returned.

Sample Input

Num 1: 23

Num 2: 563

Sample Output

0

Case 1

Input (stdin)

23 563

Output (stdout)

0

Case 2

Input (stdin)

123 463

Output (stdout)

0

Program:

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

```
int numberOfCarries(int num1 , int num2)
{
   int carry = 0, sum, p, q, count = 0;
   while((num1!=0)&&(num2!=0))
   {
      p = num1 % 10;
      q = num2 % 10;
      sum = carry + p + q;
      if(sum>9)
      {
         carry = 1;
         count++;
      }

     else
      {
         carry = 0;

      }
      num1 = num1/10;
      num2 = num2/10;
   }
   return count;
}
int main()
{
  int x, y, a;

  cin >> x >> y;

  a = numberOfCarries(x, y);
  cout << a;
  return 0;
}
```

Program 3: Replace character

Problem Statement

You are given a function,

Void *ReplaceCharacter(Char str[], int n, char ch1, char ch2);

The function accepts a string ‘ str’ of length n and two characters ‘ch1’ and ‘ch2’ as its arguments . Implement the function to modify and return the string ‘ str’ in such a way that all occurrences of ‘ch1’ in original string are replaced by ‘ch2’ and all occurrences of ‘ch2’ in original string are replaced by ‘ch1’.

Assumption: String Contains only lower-case alphabetical letters.

Note:

- Return null if string is null.
- If both characters are not present in string or both of them are same , then return the string unchanged.

Example:

Input:

Str: hello

ch1:e

ch2:o

Output:

holle

Explanation: ‘e' in original string is replaced with ‘o’ and ‘o’ in original string is replaced with ‘e’, thus output is holle.

Case 1

Input (stdin)

hello e o

Output (stdout)

holle

Case 2

Input (stdin)

tamil a i

Output (stdout)

timal

Program:

```
#include<iostream>
#include<string.h>
using namespace std;

void ReplaceCharacter(char str[], int n, char ch1, char ch2)
{
```

```
    int i;
    for(i=0; i<n ; i++)
    {
       if(str[i]==ch1)
       {
          str[i]=ch2;
       }
       else if(str[i]==ch2)
       {
          str[i]=ch1;
       }
    }

    cout << str;
}
int main()
{
    char a[100];
    char b, c;
    int len;
    cin >> a >> b >> c;
    len = strlen(a);
    ReplaceCharacter(a, len, b, c);
    return 0;
}
```

Program 4: Second greatest element

Write a program to print the second greatest element in the given array. In case, if the second largest element is not present in the array, print "There is no second largest element" and if the size of the array is less than 2, print "Invalid Input".

Sample Input:
5
12 5 7 3 90
Sample Output:
12

Case 1
Input (stdin)
5
12 5 7 3 90
Output (stdout)
12

Case 2
Input (stdin)
10
10 20 50 43 60 51 70 82 93 55
Output (stdout)
82

Program:

```
#include<iostream>
using namespace std;
#include <limits.h>

void print2largest(int arr[], int arr_size)
{
   int i, first, second;
   if (arr_size < 2)
   {
      cout<<"Invalid Input";
      return;
   }
   first = second = INT_MIN;
   for (i = 0; i < arr_size ; i ++)
   {
```

```
        if (arr[i] > first)
        {
            second = first;
            first = arr[i];
        }
        else if (arr[i] > second && arr[i] != first)
            second = arr[i];
    }

    if (second == INT_MIN)
        cout<<"There is no second largest element";
    else
        cout<<second;
}

int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++)
        cin>>arr[i];
    print2largest(arr, n);
    return 0;
}
```

Program 5: Most occurring character

Write a program to find the most occurring character in the string.

Sample Input:
Happy coding
Sample Output:
p

Case 1
Input (stdin)
Happy coding
Output (stdout)
p

Case 2
Input (stdin)
programming is good
Output (stdout)
p

Program:

```
#include<iostream>
using namespace std;
#include <string.h>

int main()
{
   char str[100], result;
   int i, len;
   int max = -1;
   int freq[256] = {0};
   scanf("%[^\n]s",str);
   len = strlen(str);
   for(i = 0; i < len; i++)
   {
      freq[str[i]]++;
   }
   for(i = 0; i < len; i++)
   {
      if(max < freq[str[i]])
      {
```

```
            max = freq[str[i]];
            result = str[i];
        }
    }

    cout<<result;
    return 0;
}
```

Program 6: Integer Difference

Given an array arr and two integer value n and m as input, take the first element and find the difference between the first element and the integer value n. If the difference is less than m then increment that particular array element. Do this for all the element in an array and print the final modified array as output.

Sample Input:
5
2 1 4 5 7
3 2
Sample Output:
3 2 5 5 7

Case 1
Input (stdin)
5
2 1 4 5 7
3 2
Output (stdout)
3 2 5 5 7

Case 2
Input (stdin)
5
1 2 3 4 5
3 5
Output (stdout)
2 3 4 5 6

Program:

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

int result (int arr[], int x, int n,int m);

int main ()
{
  int x, i, n,m;
  cin>>x;
  int arr[x];
  for (i = 0; i < x; i++)
```

```
    {
      cin>>arr[i];
    }
  cin>>n;
  cin>>m;
  result (arr, x, n, m);
  for(i=0;i<x;i++)
  {
    cout<<arr[i]<<" ";
  }
  return 0;
}

int result (int arr[], int x, int n,int m)
{
  int i;
  for (i = 0; i < x; i++)
    {
      if ( (arr[i]-n ) < m)
      {
       arr[i]=arr[i]+1;
      }
    }
}
```

Program 7: Search an element

Write a program to search an element in an array and print its index value. If the element is not present, then print -1.

Sample Input:
5
4 6 7 2 8
7
Sample Output:
index = 2

Case 1
Input (stdin)
5
4 6 7 2 8
7
Output (stdout)
index = 2

Case 2
Input (stdin)
5
4 6 3 2 8
5
Output (stdout)
-1

Program:

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

int search (int arr[], int ele, int n);

int main ()
{
  int n, i, ele;
  cin>>n;
  int arr[n];
  for (i = 0; i < n; i++)
    {
```

```
    cin>>arr[i];
   }
  cin>>ele;
  search (arr, ele, n);
  return 0;
}

int search (int arr[], int ele, int n)
{
  int i, flag = 0;

  for (i = 0; i < n; i++)
    {
     if (ele == arr[i])
    {
     cout<<"index = "<< i;
     flag = 1;
     break;
    }
    }

  if (flag == 0)
    {
     cout<<"-1";
    }
}
```

Program 8: Decimal to Binary equivalent

Write a program to convert decimal number to binary equivalent number.

Sample Input:
10
Sample Output:
1010

Case 1
Input (stdin)
10
Output (stdout)
1010

Case 2
Input (stdin)
2
Output (stdout)
10

Program:

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

int main()
{
   int a[50],n,i=0;
   cin>>n;
   while(n>0)
   {
      a[i++]=n%2;
      n=n/2;
   }
   for(i=i-1;i>=0;i--)
      cout<<a[i];
   return 0;
}
```

Program 9: Binary to decimal equivalent

Write a program to convert a binary number to an equivalent decimal value.

Sample Input:
1010
Sample Output:
10

Case 1
Input (stdin)
1010
Output (stdout)
10

Case 2
Input (stdin)
11111
Output (stdout)
31

Program:

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

int main()
{
   int  num, binary, decimal = 0, base = 1, rem;
   cin>>num;
   binary = num;
   while (num > 0)
   {
      rem = num % 10;
      decimal = decimal + rem * base;
      num = num / 10 ;
      base = base * 2;
   }
   cout<<decimal;
   return 0;
}
```

Program 10: Insert an element

Write a program to create a dynamic array and insert an element in it, in the specified position.

Input Format

First line consists of integer value N

Second line consists of an array elements

Third line consists of integer value as position and value

Output Format

Output consists of an array element

Refer the sample output for formatting

Sample Input:

5

1 2 3 4 5

2 10

Sample Output:

1 10 2 3 4 5

Case 1

Input (stdin)

5

1 2 3 4 5

2 10

Output (stdout)

1 10 2 3 4 5

Case 2

Input (stdin)

5

1 7 8 4 5

2 20

Output (stdout)

1 20 7 8 4 5

Program:

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

int main()
{
```

```
    int *array, position, i, n, value;
    cin>>n;
    array = (int*) malloc(n * sizeof(int));
    if(array == NULL)
    {
       cout<<"Error! memory not allocated.";
       exit(0);
    }
    for(i = 0; i < n; ++i)
    {
       cin>>array[i];
    }
    cin>>position>>value;

    for (i = n - 1; i >= position - 1; i--)
       array[i+1] = array[i];

    array[position-1] = value;

    for (i = 0; i <= n; i++)
       cout<<array[i]<<" ";
    return 0;
}
```

Program 11: Maximum Exponent

You are given a function,

Int MaxExponents (int a , int b);

You have to find and return the number between ‘a’ and ‘b’ (range inclusive on both ends) which has the maximum exponent of 2.

The algorithm to find the number with maximum exponent of 2 between the given range is

1. Loop between ‘a’ and ‘b’. Let the looping variable be ‘i’.
2. Find the exponent (power) of 2 for each ‘i’ and store the number with maximum exponent of 2 , let say ‘max’. Set ‘max’ to ‘i’ only if ‘i’ has more exponent of 2 than ‘max’.
3. Return ‘max’.

Assumption: a <b

Note: If two or more numbers in the range have the same exponents of 2 , return the small number.

Example

Input:

7

12

Output:

8

Explanation:

Exponents of 2 in:

7-0

8-3

9-0

10-1

11-0

12-2

Hence maximum exponent if two is of 8.

Case 1

Input (stdin)

7

12

Output (stdout)

8

Case 2

Input (stdin)

4

20

Output (stdout)
16

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

int countExp(int i){
   int count=0;
   while((i%2==0) && (i!=0)){
      count+=1;
      i=i/2;
   }
   return count;
}

int maxExp(int a, int b){
   int max = -1,num=0,temp;
   for(int i=a;i<=b ;i++){
      temp=countExp(i);
      if(temp>max){
         max=temp;
         num=i;
      }
   }
   return num;
}

int main() {
   int a,b;
   cin>>a;
   cin>>b;
   cout<<maxExp(a,b);
   return 0;
}
```

Program 12: Intermediate palindrome numbers

Write a program that it takes a lower limit and upper limit as inputs and print all the intermediate palindrome numbers.

TestCase 1:
Input :
10 80
Expected Result:
11 22 33 44 55 66 77

Test Case 2:
Input:
100 200
Expected Result:
101 111 121 131 141 151 161 171 181 191

Case 1
Input (stdin)
10
80
Output (stdout)
11 22 33 44 55 66 77

Case 2
Input (stdin)
100
200
Output (stdout)
101 111 121 131 141 151 161 171 181 191

Program:

```
#include<iostream>
using namespace std;
int reverse(int);
int main()
{
   int i,f,l;
   cin>>f;
   cin>>l;
   for (i = f; i <= l; i++)
   {
```

```
        if(i==reverse(i))
        cout<<i<<" ";
    }
    return 0;
}
int reverse(int a)
{
    int n=0,d=0,rev=0;
    n = a;
    while (n != 0)
    {
        d = n % 10;
        rev = rev * 10 + d;
        n = n / 10;
    }
    return rev;
}
```

Program 13: Delete at specific position

Write a program to create a dynamic array and delete an element from an array from the specified position. If deletion is not possible, print "Deletion not possible.".

Sample Input:

5
1 2 3 4 5
2

Sample Output:

1 3 4 5

Case 1

Input (stdin)

5
1 2 3 4 5
2

Output (stdout)

1 3 4 5

Case 2

Input (stdin)

5
1 2 3 4 5
5

Output (stdout)

1 2 3 4

Program:

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
   int *array, position, i,c, n, value;
   scanf("%d", &n);
   array = (int*) malloc(n * sizeof(int));
   if(array == NULL)
   {
      printf("Error! memory not allocated.");
      exit(0);
   }
   for(i = 0; i < n; ++i)
   {
```

```
      scanf("%d", &array[i]);
   }
   scanf("%d", &position);

   if ( position >= n+1 )
      printf("Deletion not possible.\n");

   else
   {
      for ( c = position - 1 ; c < n - 1 ; c++ )
         array[c] = array[c+1];
      for( c = 0 ; c < n - 1 ; c++ )
         printf("%d ", array[c]);
   }
   return 0;
}
```

Program 14: Segregate 0's and 1's

Write a program to segregate all the 0's in the left side and 1's in the right side in the same array.

Sample Input:
5
0 1 0 1 0
Sample Output:
0 0 0 1 1

Case 1
Input (stdin)
5
0 1 0 1 0
Output (stdout)
00011

Case 2
Input (stdin)
6
1 1 0 0 0 0
Output (stdout)
000011

Program:

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

void segregate0and1(int arr[],int n)
{
   int left = 0, right = n-1;
   while(1)
   {
      if(left >= right)
         break;
      if(arr[left] == 0)
         left++;
      else if(arr[right] == 1)
         right--;
      else
      {
         arr[left]  = 0;
```

```
            arr[right] = 1;
            left++;
            right--;
        }
    }
}

int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++)
        cin>>arr[i];
    segregate0and1(arr, n);
    for(int i=0;i<n;i++)
        cout<<arr[i];
    return 0;
}
```

Program 15: Insertion sort

Write a program to create a dynamic array and sort the elements in it using Insertion sort.

Input Format

First line consists of integer value N

Second line consists of an array element

Output Format

Output consists of an array element

Refer the sample output for formatting

Sample Input:

6

1 2 3 4 3 3

Sample Output:

1 2 3 3 3 4

Case 1

Input (stdin)

6

1 2 3 4 3 3

Output (stdout)

1 2 3 3 3 4

Case 2

Input (stdin)

7

8 1 2 5 6 4 2

Output (stdout)

1 2 2 4 5 6 8

Program:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int n, i, j, *ptr, temp;
   cin>>n;
   ptr = (int*) malloc(n * sizeof(int));
   if(ptr == NULL)
   {
```

```
        cout<<"Error! memory not allocated.";
        exit(0);
    }
    for(i = 0; i < n; ++i)
    {
        cin>>ptr[i];
    }
    for(i=1;i<n;i++)
    {
        temp = ptr[i];
        j=i-1;
        while(temp < ptr[j] && j>=0)
        {
            ptr[j+1] = ptr[j];
            --j;
        }
        ptr[j+1]=temp;
    }
    for(i=0; i<n; i++)
        cout<<ptr[i]<<" ";
    free(ptr);
    return 0;
}
```

Program 16: Sorting the first half of the array in ascending order and the second half in descending order

Write a program to sort the first half of an array in ascending order and the second half in descending order.

Input Format:
The first line contains an integer 'N', denoting the size of the array.
The next line contains space-separated integers denoting the elements of the array.
Output Format:
Print sorted array

Sample Input:
6
1 2 3 4 5 6
Sample Output:
1 2 3 6 5 4

Case 1
Input (stdin)
9
5 2 4 7 9 3 1 6 8
Output (stdout)
2 4 5 7 9 8 6 3 1

Case 2
Input (stdin)
6
10 30 20 40 60 50
Output (stdout)
10 20 30 60 50 40

Program:

```
#include<iostream>
using namespace std;
int insertion_sort(int n,int list[]);
int main()
{
   int n;
   cin>>n;
   int list[n];
   for(int index = 0;index < n; index++)
   {
```

```
        cin>>list[index];
    }
    insertion_sort(n,list);
    return 0;
}
int insertion_sort(int n,int list[])
{
    for (int idx1 = 1; idx1 < n/2; idx1++)
        {
            int key = list[idx1];
            int idx2 = idx1 - 1;
            while((idx2 >= 0) && (list[idx2] > key))
            {
                list[idx2 + 1] = list[idx2];
                idx2--;
            }
            list[idx2 + 1] = key;
        }
        for (int idx1 = n/2; idx1 < n; idx1++)
        {
            int key = list[idx1];
            int idx2 = idx1 - 1;
            while((idx2 >= n/2) && (list[idx2] < key))
            {
                list[idx2 + 1] = list[idx2];
                idx2--;
            }
            list[idx2 + 1] = key;
        }
        for(int i = 0; i < n; i++)
        {
            cout<< list[i]<<" ";
        }
    return 0;
}
```

Program 17: Regular Expression

Write a program to check the regular expression matching. Hint : '+' indicates consecutive multiple occurrence of that particular character.

Input Format

First line consists of a string s

Second line consists of string s

Output Format

Output consists of string whether it is matched or not

Refer the sample output for formatting

Sample Input:

a+b+c

aaabbc

Sample Output:

Matched

Case 1

Input (stdin)

a+b+c

aaabbc

Output (stdout)

Matched

Case 2

Input (stdin)

b+c+x

bbcxxz

Output (stdout)

Not Matched

Program:

```
#include<iostream>
using namespace std;
int main()
{
  char arr[20],ans[20];
  int i=0,j=0,len;
  cin>>arr>>ans;
  for(len=0;arr[len]!='\0';len++);
  while(arr[i]==ans[j] && arr[i]!='\0')
```

```
    {
    if(arr[i+1]=='+')
    {
      for( ; ans[j]==arr[i] ; j++);
      i=i+2;
    }
    else
    {
      i++;
      j++;
    }
   }

    if(i>=len && ans[j]=='\0')
    {
     cout<<"Matched";
    }
    else
      {
     cout<<"Not Matched";
      }
    return 0;
}
```

Program 18: Implement the following Function

def differenceofSum(n. m)

The function accepts two integers n, m as arguments Find the sum of all numbers in range from 1 to m(both inclusive) that are not divisible by n. Return difference between sum of integers not divisible by n with sum of numbers divisible by n.

Assumption:

- n>0 and m>0
- Sum lies between integral range

Example

Input

n:4

m:20

Output

90

Explanation

- Sum of numbers divisible by 4 are 4 + 8 + 12 + 16 + 20 = 60
- Sum of numbers not divisible by 4 are 1 +2 + 3 + 5 + 6 + 7 + 9 + 10 + 11 + 13 + 14 + 15 + 17 + 18 + 19 = 150
- Difference 150 – 60 = 90

Sample Input

n:3

m:10

Sample Output

19

Program:

```
#includ<stdio.h>;
int differenceofSum(int n, int m)
{
   int i, sum1 = 0, sum2 = 0;
   for(i=1; i<=m; i++)
   {
      if(i%n==0)
      {
         sum1 = sum1 + i;
      }
      else
      {
         sum2 = sum2 + i;
      }
   }
   return sum2 - sum1;
```

```
}

int main()
{
    int n, m;
    int result;
    scanf("%d",&n);
    scanf("%d",&m);
    result = differenceofSum(n, m);
    printf("%d",result);
    return 0;
}
```

Input:
3
10
Output:
19

Program 19: You are required to implement the following Function

def LargeSmallSum(arr)

The function accepts an integers arr of size 'length' as its arguments you are required to return the sum of second largest largest element from the even positions and second smallest from the odd position of given 'arr'

Assumption:

- All array elements are unique
- Treat the 0th position a seven

NOTE

- Return 0 if array is empty
- Return 0, if array length is 3 or less than 3

Example

Input

arr:3 2 1 7 5 4

Output

7

Explanation

- Second largest among even position elements(1 3 5) is 3
- Second largest among odd position element is 4
- Thus output is 3+4 = 7

Sample Input

arr:1 8 0 2 3 5 6

Sample Output

8

Program:

```
#include <stdio.h>;

int largeSmallSum(int *array, int n)
{
   int answer, i, j, temp;;
   int even[n], odd[n];
   int evencount = 0, oddcount = 0;
   if(n<=3)
   {
      answer = 0;
   }
   else
   {
```

```
    even[0] = array[0];
    evencount = 1;
    for(i=1; i<n; i++)                  //creating two array even and odd
    {
       if(i%2==0)
       {
          even[evencount] = array[i];
          evencount++;
       }
       else
       {
          odd[oddcount] = array[i];
          oddcount++;
       }
    }
    for(i=0; i<evencount; i++)          //sorting of even array
    {
       for(j=i+1; j<evencount; j++)
       {
          if(even[i]>even[j])
          {
             temp = even[i];
             even[i] = even[j];
             even[j] = temp;
          }
       }
    }
    for(i=0; i<oddcount; i++)           //sorting of odd array
    {
       for(j=i+1; j<oddcount; j++)
       {
          if(odd[i]>odd[j])
          {
             temp = odd[i];
             odd[i] = odd[j];
             odd[j] = temp;
          }
       }
    }
    answer = even[evencount-2] + odd[1];
}
```

```
    return answer;
}

int main()
{
    int n, result, i;
    scanf("%d",&n);
    int array[n];
    for(i=0; i<n; i++)
    {
        scanf("%d",&array[i]);
    }
    result = largeSmallSum(array, n);
    printf("%d",result);
    return 0;
}
```

Program 20: Implement the following Function

def ProductSmallestPair(sum, arr)

The function accepts an integers sum and an integer array arr of size n. Implement the function to find the pair, (arr[j], arr[k]) where j!=k, Such that arr[j] and arr[k] are the least two elements of array (arr[j] + arr[k] <= sum) and return the product of element of this pair

NOTE

- Return -1 if array is empty or if n<2
- Return 0, if no such pairs found
- All computed values lie within integer range

Example

Input

sum:9

Arr:5 2 4 3 9 7 1

Output

2

Explanation: Pair of least two element is (2, 1) 2 + 1 = 3 < 9, Product of (2, 1) 2*1 = 2. Thus, output is 2

Sample Input

sum:4

Arr:9 8 3 -7 3 9

Sample Output

-21

Program:

```
#include<stdio.h>;

int productSmallestPair(int *array, int n, int sum)
{
   int answer, temp, i, j, check;
   if(n<=2)
   {
      answer = -1;
   }
   else
   {
      for(i=0; i<n; i++)          //sorting of array
      {
         for(j=i+1; j<n; j++)
         {
```

```
            if(array[i]>array[j])
            {
                temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
    }
    check = array[0] + array[1];
    if(check<=sum)
    {
        answer = array[0] * array[1];
    }
    else
    {
        answer = 0;
    }
  }
  return answer;
}

int main()
{
  int n, sum, result, i;
  scanf("%d",&sum);
  scanf("%d",&n);
  int array[n];
  for(i=0; i<n; i++)
  {
    scanf("%d",&array[i]);
  }
  result = productSmallestPair(array, n, sum);
  printf("%d",result);
  return 0;
}
```

Program 21

Problem Statement

You are required to input the size of the matrix then the elements of matrix, then you have to divide the main matrix in two sub matrices (even and odd) in such a way that element at 0 index will be considered as even and element at 1st index will be considered as odd and so on. then you have sort the even and odd matrices in ascending order then print the sum of second largest number from both the matrices

Example

- **enter the size of array** : 5
- **enter element at 0 index** : 3
- **enter element at 1 index :** 4
- **enter element at 2 index** : 1
- **enter element at 3 index :** 7
- **enter element at 4 index** : 9

Sorted even array :
1 3 9
Sorted odd array :
4 7
10

Program:

```
#include <stdio.h>

int main()
{
int arr[100];
int length, i, j, oddlen, evenlen, temp, c, d;
int odd[50], even[50];

printf("enter the length of array : ");
scanf("%d",&length);

for(i=0;i<length;i++)
{
printf("Enter element at %d index : ",i);
scanf("%d",&arr[i]);
}

if(length%2==0)
{
oddlen = length/2;
```

```
evenlen = length/2;
}
else
{
oddlen = length/2;
evenlen = (length/2) + 1;
}

for(i=0;i<length;i++) // seperation of even and odd array
{
if(i%2==0)
{
even[i/2] = arr[i];
}
else
{
odd[i/2] = arr[i];
}
}

for(i=0; i<evenlen-1; i++) // sorting of even array
{
for(j=i+1; j<evenlen; j++)
{
temp = 0;
if(even[i]>even[j])
{
temp = even[i];
even[i] = even[j];
even[j] = temp;
}
}
}

for(i=0; i<oddlen-1; i++) // sorting of odd array
{
for(j=i+1; j<oddlen; j++)
{
temp = 0;
if(odd[i]>odd[j])
{
```

```
temp = odd[i];
odd[i] = odd[j];
odd[j] = temp;
}
}
}

printf("\nSorted even array : "); // printing even array
for(i=0;i<evenlen;i++)
{
printf("%d ",even[i]);
}

printf("\n");

printf("Sorted odd array : "); // printing odd array
for(i=0;i<oddlen;i++)
{
printf("%d ",odd[i]);
}

printf("\n\n%d",even[evenlen-2] + odd[1]); // printing final result
}
```

Program 22

In this lockdown a family of N members decided to play a game the rules of which are :-

- All *N* members are made to sit uniformly in a circle (ie. from *1* to *N* in clockwise direction).
- The game start with the person sitting at first position.
- A song is played in the background. The lyrics of the song are denoted by a string which consists of only letters 'x' and 'y'. Assume that each lyric of the song is a single letter.
- If the lyric 'x' occurs in the song, the member who is currently holding the Parcel passes it on to the next member. This passing takes place in clockwise direction.
- If the lyric 'y' occurs in the song, the member who is currently holding the Parcel loses his/her chances of winning the game. He/she hands over the parcel to the next member (in clockwise direction) and moves out.
- The game continues until a single member survives in the end. He/she will be the winner of the game.
- Note that the song repeats continuously ie. while the game is going on, if at all the song ends, the stereo system will automatically start playing the song from the start without any delay.

You have to find out the member who wins the game.

Input: The input consists of *2* lines. The first line consists of *N*, the member of family in the class. The next line consists of a string denoting the lyrics of the song the teacher plays.
Output: Print a single integer denoting the roll number of the student who wins the game.

Constraints:
$2 \leq N \leq 100000$
$1 \leq |S| \leq 10000$, where $|S|$ denotes the length of the input string. It is guaranteed that at least *1* lyric in the song will be a 'y'

Sample Input:
3
xyx
Sample Output:
1

Explanation:
Starting from 1 lyrics : 'x' therefore he passes the ballto 2nd
2nd turn lyrics : 'y' therefore 2nd member gets out of game and passes to 3rd
3rd turn lyrics : 'x' therefore 3rd passes ball to first.
4th turn lyrics : 'x' passes to 3rd
5th turn lyrics: 'y' therefore gets eliminated.
Hence person sitting at position 1 won this game.

Test Case 1
Input
3 xyx

Output
1

Test Case 2
Input
6 xxyyxy
Output
2

Program:
```
#include<bits/stdc++.h>
#include<string>
using namespace std;
#define ll long long
ll n,slen;

void fun (ll size,ll k,string s, ll l, ll *ingame)
{
   if(size==1)
   {
      ll i=1;
      while(ingame[i]!=1)
      {i++;

      }
      cout<<i<<endl;
      return;
   }
   if(l==slen) l=0;
   if(k==n+1)  k=1;
   if(ingame[k]==0)
   {

      fun(size,k+1,s,l,ingame);
   }
   else
   {
     if(s[l]=='x')
     fun(size,k+1,s,l+1,ingame);
     else{
        ingame[k]=0;
```

```
            fun(size-1,k+1,s,l+1,ingame);
        }
    }
    return;
}

int main()
{
    cin>>n;
    string s;
    cin>>s;
    slen=s.length();
    ll ingame[n+1];
    for(int i =1;i<=n;i++)
    {
        ingame[i]=1;
    }
    fun(n,1,s,0,ingame);
}
```

Program 23:

Write a program to find the equilibrium element in the given array.
Hint : A[i] is equilibrium element if A[0] + A[1] + … A[i-1] == A[i+1] + A[i+2] + … + A[n-1]

Sample Input 1:
6
1 2 3 4 3 3
Sample Output 1:
4

Sample Input 2:
6
5 6 4 2 3 1
Sample Output 2:
No Equilibrium element found

Test Case 1
Input
6 1 2 3 4 3 3
Output
4

Test Case 2
Input
6 5 6 4 2 3 1
Output
No Equilibrium element found

Program:

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

int findElement(int arr[], int n)
{
    int prefixSum[n];
    prefixSum[0] = arr[0];
    for (int i = 1; i < n; i++)
        prefixSum[i] = prefixSum[i - 1] + arr[i];
    int suffixSum[n];
    suffixSum[n - 1] = arr[n - 1];
    for (int i = n - 2; i >= 0; i--)
```

```
        suffixSum[i] = suffixSum[i + 1] + arr[i];
    for (int i = 1; i < n - 1; i++)
        if (prefixSum[i] == suffixSum[i])
            return arr[i];
    return -1;
}

int main()
{
    int n;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++)
        cin>>arr[i];
    int result = findElement(arr, n);
    if(result == -1)
        cout<<"No Equilibrium element found";
    else
        cout<<result;
    return 0;
}
```

Program 24: Move hyphen to the beginning

Implement the following function: char* MoveHyphen(char str[], int n); The function accepts a string ‘str’ of length ‘n’, that contains alphabets and hyphens (-). Implement the function to move all hyphens (-) in the string to the front of the given string. Note: Return null if str is null.

Input Format: Given input is in given string
Output Format: Output is in string format

Refer the sample output for formatting
Sample Input:
Move-Hyphens-to-Front
Sample Output:
---MoveHyphenstoFront

Explanation: The string “Move-Hyphens-to-Front” has 3 hyphens(-) which are moved to the front of the string thus output is “---MoveHyphenstoFront”.

Case 1
Input (stdin)
Move-Hyphens-to-Front

Output (stdout)
---MoveHyphenstoFront

Case 2
Input (stdin)
Hello-world-is

Output (stdout)
--Helloworldis

Program 25: Non repeating elements

Write a program to eliminate the common elements in the given 2 arrays and print only the non repeating elements and the total.

Sample Input:

5 4
1 2 8 6 5
2 6 8 10

Sample Output:

1 5 10
3

Case 1

Input (stdin)

5 4
1 2 8 6 5
2 6 8 10

Output (stdout)

1 5 10
3

Case 2

Input (stdin)

5 5
10 20 30 40 50
60 70 80 90 50

Output (stdout)

10 20 30 40 60 70 80 90
8

Category: Difficult

Program 1

Write a program to check the regular expression matching. Hint : '+' indicates consecutive multiple occurrence of that particular character.

Input Format

First line consists of a string s
Second line consists of string s

Output Format

Output consists of string whether it is matched or not

Refer the sample output for formatting

Sample Input:

a+b+c
aaabbc

Sample Output:

Matched

Test Case 1

Input
a+b+c
aaabbc
Output
Matched

Test Case 2

Input
b+c+x
bbcxxz
Output
Not Matched

Program:

```
#include<iostream>
using namespace std;
int main()
{
  char arr[20],ans[20];
  int i=0,j=0,len;
  cin>>arr>>ans;
```

```
  for(len=0;arr[len]!='\0';len++);
  while(arr[i]==ans[j] && arr[i]!='\0')
   {
   if(arr[i+1]=='+')
   {
     for( ; ans[j]==arr[i] ; j++);
     i=i+2;
   }
   else
   {
     i++;
     j++;
   }
  }

   if(i>=len && ans[j]=='\0')
   {
    cout<<"Matched";
   }
   else
     {
    cout<<"Not Matched";
     }
   return 0;
}
```

Program 2: Find the Nth element

Write a program to create a Singly Linked List and find the Nth element from the end of the list.
Hint: Insertion at the beginning.

Output: output consists of single line, print the element in the given position in a list, else print "**No node found**"

Sample Input:
5
10 8 6 4 0
2
Output:
8

Case 1
Input (stdin)
5
10 8 6 4 0
2
Output (stdout)
8

Case 2
Input (stdin)
10
100 200 300 400 500 600 700 800 900 101
11
Output (stdout)
No node found

Program:

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct Node
{   int num;
   struct Node* link;
};
void insertAtBeginning(struct Node** head_ref, int new_data)
{
   struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
   new_node->num = new_data;
```

```
    new_node->link = (*head_ref);
    (*head_ref) = new_node;
}
int printNthFromLast(struct Node* head, int n)
{
      int len = 0, i;
    struct Node *temp = head;
    while (temp != NULL)
    {
        temp = temp->link;
        len++;
    }
    if (len < n)
        return 0;
    temp = head;
    for (i = 1; i < len-n+1; i++)
        temp = temp->link;
        cout<<temp->num;
    return 1;
}
int main()
{
      struct Node* head = NULL;
      int total_elem_to_insert, tmp, i,n;
      cin>>total_elem_to_insert;

    for(i=0;i<total_elem_to_insert;i++)
    {
        cin>>tmp;
        insertAtBeginning(&head, tmp);
    }
    scanf("%d",&n);

    if(0==(printNthFromLast(head, n)))
        cout<<"No node found"<<endl;
    return 0;
}
```

Program 3: Anagram

Write a program to find whether the given string is the anagram of the first string.

Sample Input 1:
eat
ate
Sample Output 1:
Anagram

Sample Input 2:
eat
abc
Sample Output 2:
Not Anagram

Case 1
Input (stdin)
eat
ate
Output (stdout)
Anagram

Case 2
Input (stdin)
eat
abc
Output (stdout)
Not Anagram

Program:

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

int find_anagram(char [], char []);
int main()
{
   char array1[100], array2[100];
   int flag;
   cin>>array1;
   cin>>array2;
   flag = find_anagram(array1, array2);
```

```
    if (flag == 1)
        cout<<"Anagram";
    else
        cout<<"Not Anagram";
    return 0;
}
int find_anagram(char array1[], char array2[])
{
    int num1[26] = {0}, num2[26] = {0}, i = 0;
    while (array1[i] != '\0')
    {
        num1[array1[i] - 'a']++;
        i++;
    }
    i = 0;
    while (array2[i] != '\0')
    {
        num2[array2[i] -'a']++;
        i++;
    }
    for (i = 0; i < 26; i++)
    {
        if (num1[i] != num2[i])
            return 0;
    }
    return 1;
}
```

Program 4: Append without duplicates

John assigned a roll number to each student in ascending order. Every time he assigns a number to a student he wants to check whether the number is already assigned to any other student to avoid the duplicates in the roll number. Add the number to the list only if the number is not present already. Implement this concept using a Linked List. If the list is empty, print "List is empty".

Input Format: The input consists of a list of integers, negative value indicates the end of the linked list.

Output Format: The output should be numbers in the list in separate line.

Refer the sample input & output for formatting specifications.

Sample Input

11
22
33
22
33
44
-77

Sample Output

11
22
33
44

Case 1

Input (stdin)

11
22
33
22
33
44
-77

Output (stdout)

11
22
33
44

Case 2

Input (stdin)

-5

Output (stdout)
List is empty

Program:

```
#include<iostream>
using namespace std;
class Node
{
   public:
      int data;
      Node *next;
};
void append(Node **headadd,int data)
{
   Node *temp,*newnode,*temp1;
   temp = *headadd;
   temp1 = *headadd;
   int flag = 1;
   while(temp1 != NULL)
   {
      if(temp1->data == data)
         flag = 0;
      temp1 = temp1->next;
   }
   if(flag == 1)
   {
      newnode = new Node();
      newnode->data = data;
      newnode->next = NULL;
      if(*headadd == NULL)
         *headadd = newnode;
      else
      {
         while(temp->next != NULL)
            temp = temp->next;
         temp->next = newnode;
      }
   }
}
void display(Node *head)
{
```

```
    if(head == NULL)
    {
        cout<< "List is empty";
        return;
    }
    while(head != NULL)
    {
        cout<<head->data<<endl;
        head = head->next;
    }
}
int main()
{
    Node *head = NULL;
    int data;
    do
    {
        cin>>data;
        if(data>0)
            append(&head,data);
    }while(data>0);
    display(head);
    return 0;
}
```

Program 5: Merge two sorted linked list

Write a program to merge two sorted linked lists. Given two sorted linked lists say list1 and list2, merge list2 with list1. Insertion has to be done at the end of the list.

Input Format

The first line consists of the size of the first sorted linked list.
The second line consists of list of elements of first sorted linked list separated by space.
The third line consists of the size of the second sorted linked list.
The fourth line consists of list of elements of second sorted linked list separated by space.

Output Format: The output should be the merged list of two linked list.

Refer the sample input & output for formatting specifications.

Sample Input

3
1 3 5
3
2 4 6

Sample Output

1->2->3->4->5->6->NULL

Case 1

Input (stdin)

3
1 3 5
3
2 4 6

Output (stdout)

1->2->3->4->5->6->NULL

Program 6

N-base notation is a system for writing numbers which uses only n different symbols, This symbols are the first n symbols from the given notation list(Including the symbol for o) Decimal to n base notation are (0:0, 1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8, 9:9, 10:A,11:B and so on upto 35:Z)

Implement the following function

Char* DectoNBase(int n, int num):

The function accept positive integer n and num Implement the function to calculate the n-base equivalent of num and return the same as a string

Steps:

1. Divide the decimal number by n,Treat the division as the integer division
2. Write the the remainder (in n-base notation)
3. Divide the quotient again by n, Treat the division as integer division
4. Repeat step 2 and 3 until the quotient is 0
5. The n-base value is the sequence of the remainders from last to first

Assumption: 1 < n < = 36

Example

Input

n: 12

num: 718

Output

4BA

Explanation:

num	Divisor	quotient	remainder
718	12	59	10(A)
59	12	4	11(B)
4	12	0	4(4)

Sample Input

n: 21

num: 5678

Sample Output

CI8