Write a function SmallLargeSum(array) which accepts the array as an argument or parameter, that performs the addition of the second largest element from the even location with the second largest element from an odd location?
Rules:
a. All the array elements are unique.
b. If the length of the array is 3 or less than 3, then return 0.
c. If Array is empty then return zero.

Sample Test Case 1:
Input:
6
3 2 1 7 5 4
Output:
7

Explanation: The second largest element in the even locations (3, 1, 5) is 3. The second largest element in the odd locations (2, 7, 4) is 4. So the addition of 3 and 4 is 7. So the answer is 7.

Sample Test Case 2:
Input:
7
4 0 7 9 6 4 2
Output:
10

Solution:

- First we've to take input from the user i.e in the main function.
- Then we've to create two different arrays in which the first array will contain all the even position elements and the second one will contain odd position elements.
- The next step is to sort both the arrays so that we'll get the second-largest elements from both the arrays.
- At last, we've to add both the second-largest elements that we get from both the arrays.
- Display the desired output in the main function.

Code Implementation:

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

```
using namespace std;

int smallLargeSum(int *arr, int n) {

    if(n <= 3) {

        return 0;

    }

    //Here we use vector because we don't know the array size,

    //we can use array also but vector gives us more functionality than
array

    vector<int> arrEven, arrOdd;

    //Break array into two different arrays even and odd

    for(int i = 0; i < n; i++) {

        //If Number is even then add it into even array

        if(i % 2 == 0) {

            arrEven.push_back(arr[i]);

        }

        else {

            arrOdd.push_back(arr[i]);

        }

    }

    //Sort the even array

    sort(arrEven.begin(), arrEven.end());

    //We use sort function from C++ STL library

    //Sort the odd array

    sort(arrOdd.begin(), arrOdd.end());
```

```
    //Taking second largest element from both arrays and add them

    return arrEven[1] + arrOdd[1];

}

// Driver code

int main()

{

    int n;

    cout<<"Enter How many elements you want to enter?\n";

    cin>>n;

    int arr[n];

    //Get input from user

    cout<<"Start entering the numbers\n";

    for(int i = 0; i < n; i++) {

        cin>>arr[i];

    }

    cout<<"Output is\n";

    cout<<smallLargeSum(arr, n);

    return 0;

}
```

Write a function CheckPassword(str) which will accept the string as an argument or parameter and validates the password. It will return 1 if the conditions are satisfied else it'll return 0?
The password is valid if it satisfies the below conditions:
a. It should contain at least 4 characters.
b. At least 1 numeric digit should be present.
c. At least 1 Capital letter should be there.

d. Passwords should not contain space or slash(/).
e. The starting character should not be a number.

Sample Test Case:
Input:
bB1_89

Output:
1

Approach:

1. Using if condition check whether the length of string is greater than equal to 4 or not.
2. Run a loop on the string and check if any character is a digit or not. It is a digit if it's between '0' and '9'.
3. Iterate the string and check that only at least one letter should be between 'A' and 'Z', i.e it should be capital.
4. Run a loop on string and check no character should match space(' ') or slash ('/').
5. Check that the first character should not lie between '0' and '9'.

Code Implementation

```
#include<iostream>

#include<string.h>

using namespace std;

int CheckPassword(char str[]) {

        int len = strlen(str);

        bool isDigit = false, isCap = false,
isSlashSpace=false,isNumStart=false;

        //RULE 1: At least 4 characters in it

        if (len < 4)

            return 0;
```

```
        for(int i=0; i<len; i++) {

            //RULE 2: At least one numeric digit in it

            if(str[i]>='0' && str[i]<='9') {

        isDigit = true;

            }

            //RULE 3: At least one Capital letter

            else if(str[i]>='A'&&str[i]<='Z'){

        isCap=true;

            }

            //RULE 4: Must not have space or slash

            if(str[i]==' '|| str[i]=='/')

        isSlashSpace=true;

        }

        //RULE 5: Starting character must not be a number

        isNumStart = (str[0]>='0' && str[0]<='9');

        //FYI: In C++, if int data type function returns the true then it
prints 1 and if false then it prints 0

        return isDigit && isCap && !isSlashSpace && !isNumStart;

    }

int main() {

        char password[100];

        cout<<"Enter the Password\n";

        cin>>password;

        cout<<"The output is =\n";
```

```
        cout<<CheckPassword(password);
```

```
}
```

Write a function CalculateBinaryOperations(str) that accepts the string as an argument or parameter. The string should contain the binary numbers with their operators OR, AND, and XOR?
a. A Means the AND Operation.
b. B Means the OR Operation.
c. C Means the XOR Operation.
By scanning the given string from left to right you've to calculate the string and by taking one operator at a time then return the desired output.
Conditions:
1.The priority of the operator is not required.
2.The length of the string is always Odd.
3.If the length of the string is null then return -1.

Sample Test Case:
Input:
1C0C1C1A0B1
Output:
1

Explanation:
The entered input string is 1 XOR 0 XOR 1 XOR 1 AND 0 OR 1.
Now calculate the string without an operator priority and scan the string characters from left to right. Now calculate the result and return the desired output.
Note: This will convert the char into the num (char – '0') in the c++ language.

Code Implementation

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

using namespace std;

int CalculateBinaryOperations(char* str)

{

    int len = strlen(str);
```

```
    //Let's consider the first element as a answer (because string can be a
single char)

    int ans = str[0]-'0';

    for(int i=1; i<len-1; i+=2)

    {

        int j=i+1;

        //Performing operation for AND

        if(str[i]=='A')

        {

            ans = ans & (str[j]-'0');

        }

        //Performing operation for OR

        else if(str[i]=='B')

        {

            ans = ans | (str[j]-'0');

        }

        //Performing operation for XOR

        else if(str[i]=='C')

        {

            ans = ans ^ (str[j]-'0');

        }

    }

    return ans;

}
```

```
int main()

{

    char str[100];

    cout<<"Enter the String:\n";

    cin>>str;

    cout<<"The output is :\n";

    cout<<CalculateBinaryOperations(str);

}
```

4. **Write a function FindMaxInArray, which will find the greatest number from an array with its desired index? The greatest number and its desired index should be printed in separate lines.**

Sample Test Case:
Input:
10
15 78 96 17 20 65 14 36 18 20

Output:
96
2

```
#include<iostream>

using namespace std;

void FindMaxInArray(int arr[],int length)

{

   int max=-1, maxIdx=-1;

   for(int i = 0;i < length; i++)

   {

      if(arr[i] > max)
```

```
      {
         max = arr[i];
         maxIdx = i;
      }
   }
   cout<<"The Maximum element in an array is = \n";
   cout<<max;
   cout<<"\nAt the Index = \n";
   cout<<maxIdx;
}
int main()
{
   int n;
   cout<<"How many elements you want to enter:\n";
   cin>>n;
   int a[n];
   cout<<"Enter elements: \n";
   for(int i=0;i<n;i++)
      cin>>a[i];
   FindMaxInArray(a,n);
}
```

5. **Write a function OperationChoices(c, a, b) which will accept three integers as an argument, and the function will return:**
a. (a + b) if the value of c=1.

b. (a – b) if the value of c=2.
*c. (a b) if the value of c=3.
d. (a / b) if the value of c=4.**

Sample Test Case:
Input:
2
15
20

Output:
-5

Explanation:
Here, the value of the c is two i.e 2. So it'll perform the operation as subtraction (15, 20) and will return -5.

Code Implementation

```
#include<iostream>

using namespace std;

int operationChoices(int c, int a , int b)

{

    if(c==1)

    {

        return a + b;

    }

    else if(c==2)

    {

        return a - b;

    }

    else if(c==3)
```

```
    {
        return a * b;
    }
    else if(c==4)
    {
        return a / b;
    }
}
int main()
{
    int x, y, z;
    int result;
    cout<<"Enter c\n";
    cin>>x;
    cout<<"Enter two elements\n";
    cin>>y;
    cin>>z;
    result = operationChoices(x, y, z);
    cout<<"The result is ";
    cout<<result;
}
```

The function accepts two positive integers ‘r’ and ‘unit’ and a positive integer array ‘arr’ of size ‘n’ as its argument ‘r’ represents the number of rats present in an area, ‘unit’ is the amount of food each rat consumes and each ith element of array ‘arr’ represents the amount of food present in ‘i+1’ house number, where 0 <= i.
Note:
1.Return -1 if the array is null
2.Return 0 if the total amount of food from all houses is not sufficient for all the rats.
3.Computed values lie within the integer range.

Example:
Input:
r: 7
unit: 2
n: 8
arr: 2 8 3 5 7 4 1 2

Output:
4

Explanation:
Total amount of food required for all rats = r *unit*
= *7* 2 = 14.
The amount of food in 1st houses = 2+8+3+5 = 18. Since, the amount of food in 1st 4 houses is sufficient for all the rats. Thus, output is 4.

Code Implementation

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

using namespace std;




int calculate (int r, int unit, int arr[], int n)

{

    if (n == 0)

        return -1;
```

```
    int totalFoodRequired = r * unit;
    int foodTillNow = 0;
    int house = 0;

    for (house = 0; house < n; ++house)
    {
        foodTillNow += arr[house];
        if (foodTillNow >= totalFoodRequired)
        {
            break;
        }
    }
    if (totalFoodRequired > foodTillNow)
        return 0;
    return house + 1;
}

int main ()
{
    int r;
    cin >> r;
    int unit;
    cin >> unit;
```

```
    int n;
    cin >> n;
    int arr[n];


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

In Java

```
import java.util.*;
class Main
{
    public static int solve (int r, int unit, int arr[], int n)
    {
        if (arr == null)
            return -1;
        int res = r * unit;
        int sum = 0;
        int count = 0;
        for (int i = 0; i < n; i++)
```

```
        {
            sum = sum + arr[i];
            count++;
            if (sum >= res)
                break;
        }
        if(sum<res)
            return 0;
        return count;
    }

    public static void main (String[]args)
    {
        Scanner sc = new Scanner (System.in);
        int r = sc.nextInt ();
        int unit = sc.nextInt ();
        int n = sc.nextInt ();
        int arr[] = new int[n];

        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt ();
        System.out.println (solve (r, unit, arr, n));
    }
```

```
}
```

In Python

```
def calculate(r,unit,arr,n):
    if n==0:
        return -1

    totalFoodRequired=r*unit
    foodTillNow=0
    house=0
    for house in range(n):
        foodTillNow+=arr[house]
        if foodTillNow >= totalFoodRequired:
            break
    if totalFoodRequired > foodTillNow:
        return 0

    return house+1
r = int(input())
unit = int(input())
n = int(input())
arr = list(map(int,input().split()))
print(calculate(r,unit,arr,n))
```

You are given a function,
int findCount(int arr[], int length, int num, int diff);

The function accepts an integer array ‘arr’, its length and two integer variables ‘num’ and ‘diff’. Implement this function to find and return the number of elements of ‘arr’ having an absolute difference of less than or equal to ‘diff’ with ‘num’.
Note: In case there is no element in ‘arr’ whose absolute difference with ‘num’ is less than or equal to ‘diff’, return -1.

Example:
Input:
arr: 12 3 14 56 77 13
num: 13
diff: 2

Output:
3

Explanation:
Elements of ‘arr’ having absolute difference of less than or equal to ‘diff’ i.e. 2 with ‘num’ i.e. 13 are 12, 13 and 14.

Code Implementation

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

using namespace std;



int findCount(int n, int arr[], int num, int diff) {

    int count = 0;

    for (int i = 0; i < n; ++i)

    {

        if (abs(arr[i] - num) <= diff)

        {

            count++;
```

```
        }
    }
    return count > 0 ? count : -1;
}
int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }
    int num; cin >> num;
    int diff; cin >> diff;
    cout << findCount(n, arr, num, diff);
}
```

```
import java.util.*;
class Main
{
    public static int findCount (int arr[], int length, int num, int diff)
    {
        int count = 0;
        for (int i = 0; i < length; i++)
        {
```

```
            if (Math.abs (num - arr[i]) <= diff)
                count++;
        }
        return count>0?count:-1;
    }

    public static void main (String[]args)
    {
        Scanner sc = new Scanner (System.in);
        int n = sc.nextInt ();
        int arr[] = new int[n];

        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt ();
        int num = sc.nextInt ();
        int diff = sc.nextInt ();

        System.out.println (findCount (arr, n, num, diff));
    }
}
def findCount(n, arr, num, diff):
    count=0
    for i in range(n):
```

```
        if(abs(arr[i]-num)<=diff):
            count+=1
    if count:
        return count
    return 0
n=int(input())
arr=list(map(int,input().split()))
num=int(input())
diff=int(input())
print(findCount(n, arr, num, diff))
```

N-base notation is a system for writing numbers that 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:

- Divide the decimal number by n,Treat the division as the integer division
- Write the the remainder (in n-base notation)
- Divide the quotient again by n, Treat the division as integer division
- Repeat step 2 and 3 until the quotient is 0
- 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 = 718, divisor = 12, quotient=59, remainder=10(A).
- num = 59, divisor = 12, quotient=4, remainder=11(B).
- num = 4, divisor = 12, quotient=0, remainder=4(A).

Sample Input
n: 21
num: 5678

Sample Output
CI8

Code Implementation

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

using namespace std;

string decitoNBase (int n, int num)

{

        string res = "";

        int quotient = num / n;



        vector<int> rem;



        rem.push_back(num % n);
```

```
        while(quotient != 0)
        {
            rem.push_back(quotient % n);
            quotient = quotient / n;
        }

        for (int i = 0; i < rem.size (); i++)
        {
            if (rem[i] > 9)
            {
                res = (char)(rem[i] - 9 + 64) + res;
            }
            else
                res = to_string(rem[i]) + res;
        }

        return res;
}

int main ()
{
```

```
  int n, num;

  cin >> n>>num;



  cout << decitoNBase(n, num);



  return 0;

}
```

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 and 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
Sum = 7

Code Implementation

```
#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[oddlen-2]);    // printing
final result

}
```

10. You are required to implement the following function:
int Calculate(int m, int n);

The function accepts 2 positive integers ‘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
510

Code Implementation

```
#include <stdio.h>



int Calculate (int, int);

int main ()

{

    int m, n, result;

    // Getting Input

    printf ("Enter the value of m : ");

    scanf ("%d", &m);

    printf ("Enter the value of n : ");

    scanf ("%d", &n);



    result = Calculate (n, m);

    // Getting Output



    printf ("%d", result);

    return 0;

}



int Calculate (int n, int m)

{

    // Write your code here
```

```
    int i, sum = 0;

    for (i = m; i <= n; i++)

    {

        if ((i % 3 == 0) && (i % 5 == 0))

        {

            sum = sum + i;

        }

    }

    return sum;

}
```