There is a JAR full of candies for sale at a mall counter. JAR has the capacity N, that is JAR can contain maximum N candies when JAR is full. At any point of time. JAR can have M number of Candies where M<=N. Candies are served to the customers. JAR is never remain empty as when last k candies are left, JAR is refilled with new candies in such a way that JAR get full.  
Write a code to implement above scenario. Display JAR at counter with available number of candies. Input should be the number of candies one customer can order at point of time. Update the JAR after each purchase and display JAR at Counter.

Output should give number of Candies sold and updated number of Candies in JAR.

If Input is more than candies in JAR, return: “INVALID INPUT”

**Given,**

N=10, where N is NUMBER OF CANDIES AVAILABLE

K =< 5, where k is number of minimum candies that must be inside JAR ever.

**Example 1:(N = 10, k =< 5)**

* **Input Value**
  + 3
* **Output Value**
  + NUMBER OF CANDIES SOLD : 3
  + NUMBER OF CANDIES AVAILABLE : 7

**Example : (N=10, k<=5)**

* **Input Value**
  + 0
* **Output Value**
  + INVALID INPUT
  + NUMBER OF CANDIES LEFT : 10

#include<stdio.h>

int main()

{

int n=10, k=5;

int num;

scanf("%d",&num);

if(num>=1 && num<=10&& n-num>=0)

{

printf("NUMBER OF CANDIES SOLD : %d\n",num);

printf("NUMBER OF CANDIES LEFT : %d",n-num>k? n-num:10);

}

else

{

printf("INVALID INPUT\n");

printf("NUMBER OF CANDIES LEFT : %d",n);

}

return 0;

}

Selection of MPCS exams include a fitness test which is conducted on ground. There will be a batch of 3 trainees, appearing for running test in track for 3 rounds. You need to record their oxygen level after every round. After trainee are finished with all rounds, calculate for each trainee his average oxygen level over the 3 rounds and select one with highest oxygen level as the most fit trainee. If more than one trainee attains the same highest average level, they all need to be selected.

**Display the most fit trainee (or trainees) and the highest average oxygen level.**

**Note:**

* **The oxygen value entered should not be accepted if it is not in the range between 1 and 100.**
* If the calculated maximum average oxygen value of trainees is below 70 then declare the trainees as unfit with meaningful message as “All trainees are unfit.
* Average Oxygen Values should be rounded.

**Example 1:**

* **INPUT VALUES**

            95

            92

            95

            92

            90

            92

            90

            92

            90

* **OUTPUT VALUES**
  + Trainee Number : 1
  + Trainee Number : 3

**Note:**

Input should be 9 integer values representing oxygen levels entered in order as

**Round 1**

* Oxygen value of trainee 1
* Oxygen value of trainee 2
* Oxygen value of trainee 3

**Round 2**

* Oxygen value of trainee 1
* Oxygen value of trainee 2
* Oxygen value of trainee 3

**Round 3**

* Oxygen value of trainee 1
* Oxygen value of trainee 2
* Oxygen value of trainee 3

**Output must be in given format as in above example. For any wrong input final output should display “INVALID INPUT”**

#include <stdio.h>

int main()

{

int o,o2,o3;

float average[3] = {0};

int i, j, max=0;

for(i=0; i<3; i++)

{

scanf("%d",&o);

if(o<1 || o>100)

break ;

else

average[0]+=o;

scanf("%d",&o2);

if(o2<1 || o2>100)

break;

else

average[1]+=o2;

scanf("%d",&o3);

if(o3<1 || o3>100)

break;

else

average[2]+=o3;

}

if(i<4)

printf("INVALID INPUT");

else

{

if(average[0]+average[1]+average[2])/9 <70.0f)

printf("All trainees are unfit");

else

{

average[0]/=3;

average[1]/=3;

average[2]/=3;

if(average[0]>average[1])

{

if(average[0]>average[2])

printf("Trainee Number : 1");

else if(average[0]==average[2])

printf("Trainee Number : 1\nTrainee Number : 3");

else if (average[0]<average[2])

printf("Trainee Number : 3");

}

else if(average[0]==average[1])

{

if(average[0]>average[2])

printf("Trainee Number : 1\nTrainee Number : 1");

else if(average[0]==average[2])

printf("Trainee Number : 1\nTrainee Number :2\nTrainee Number : 3");

else if (average[0]<average[2])

printf("Trainee Number : 3");

}

else if(average[0]<average[1])

{

if(average[1]>average[2])

printf("Trainee Number : 2");

else if(average[1]==average[2])

printf("Trainee Number : 2\nTrainee Number : 3");

else if (average[1]<average[2])

printf("Trainee Number : 3");

}

}

}

return 0;

}

A washing machine works on the principle of Fuzzy System, the weight of clothes put inside it for washing is uncertain but based on weight measured by sensors, it decides time and water level which can be changed by menus given on the machine control area.

For low level water, the time estimate is 25 minutes, where approximately weight is between 2000 grams or any nonzero positive number below that.

For medium level water, the time estimate is 35 minutes, where approximately weight is between 2001 grams and 4000 grams.

For high level water, the time estimate is 45 minutes, where approximately weight is above 4000 grams.

Assume the capacity of machine is maximum 7000 grams

Where approximately weight is zero, time estimate is 0 minutes.

Write a function which takes a numeric weight in the range [0,7000] as input and produces estimated time as output is: “OVERLOADED”, and for all other inputs, the output statement is

“INVALID INPUT”.

Input should be in the form of integer value –

Output must have the following format –

Time Estimated: Minutes

**Example:**

* Input value

2000

* Output value

Time Estimated: 25 minutes

#include<stdio.h>

void calculateTime(int n)

{

if(n==0)

printf("Time Estimated : 0 Minutes");

else if(n>0 && n<=2000)

printf("Time Estimated : 25 Minutes");

else if(n>2000 && n<=4000)

printf("Time Estimated : 35 Minutes");

else if(n>4000 && n<=7000)

printf("Time Estimated : 45 Minutes");

else

printf("INVALID INPUT");

}

int main()

{

int machineWeight;

scanf("%d",&machineWeight);

calculateTime(machineWeight);

return 0;

}

The Caesar cipher is a type of substitution cipher in which each alphabet in the plaintext or messages is shifted by a number of places down the alphabet.  
For example,with a shift of 1, P would be replaced by Q, Q would become R, and so on.  
To pass an encrypted message from one person to another, it is first necessary that both parties have the ‘Key’ for the cipher, so that the sender may encrypt and the receiver may decrypt it.  
Key is the number of OFFSET to shift the cipher alphabet. Key can have basic shifts from 1 to 25 positions as there are 26 total alphabets.  
As we are designing custom Caesar Cipher, in addition to alphabets, we are considering numeric digits from 0 to 9. Digits can also be shifted by key places.  
For Example, if a given plain text contains any digit with values 5 and keyy =2, then 5 will be replaced by 7, “-”(minus sign) will remain as it is. Key value less than 0 should result into “INVALID INPUT”

**Example 1:**Enter your PlainText: All the best  
Enter the Key: 1

The encrypted Text is: Bmm uif Cftu

Write a function CustomCaesarCipher(int key, String message) which will accept plaintext and key as input parameters and returns its cipher text as output.

#include   
int main()  
{  
    char str[100];  
    int key, i=0, left;  
    printf("Enter your plain text : ");  
    scanf("%[^\n]s",str);  
    printf("Enter the key : ");  
    scanf("%d",&key);  
    if(key==0)  
    {  
        printf("INVALID INPUT");  
    }  
    else  
    {  
        while(str[i]!='\0')  
        {  
            //printf("%d\n", str[i]);  
            if(str[i]>=48 && str[i]<=57)  
            {  
                if(str[i]+key<=57)  
                {  
                    str[i] = str[i] + key;  
                }  
                else  
                {  
                    left = (str[i] + key) - 57;  
                    str[i] = 47 + left;  
                }     
            }  
            else if(str[i]>=65 && str[i]<=90)  
            {  
                if(str[i]+key<=90)  
                {  
                    str[i] = str[i] + key;  
                }  
                else  
                {  
                    left = (str[i] + key) - 90;  
                    str[i] = 64 + left;  
                }    
            }  
            else if(str[i]>=97 && str[i]<=122)  
            {  
                if(str[i]+key<=122)  
                {  
                    str[i] = str[i] + key;  
                }  
                else  
                {  
                    left = (str[i] + key) - 122;  
                    str[i] = 96 + left;  
                }   
            }  
            i++;  
        }  
        printf("The encrypted text is : %s",str);  
   }  
   return 0;  
}

**We want to estimate the cost of painting a property. Interior wall painting cost is Rs.18 per sq.ft. and exterior wall painting cost is Rs.12 per sq.ft.**

**Take input as**  
1. Number of Interior walls  
2. Number of Exterior walls  
3. Surface Area of each Interior 4. Wall in units of square feet  
Surface Area of each Exterior Wall in units of square feet

**If a user enters zero  as the number of walls then skip Surface area values as User may don’t  want to paint that wall.**

**Calculate and display the total cost of painting the property  
Example 1:**

6  
3  
12.3  
15.2  
12.3  
15.2  
12.3  
15.2  
10.10  
10.10  
10.00  
Total estimated Cost : 1847.4 INR

Note: Follow in input and output format as given in above example

#include<stdio.h>  
int main()  
{  
    int ni,ne,i=0;  
    float int\_p=18,ext\_p=12,cost=0,temp;  
    scanf("%d %d",&ni,&ne);  
    if(ni<0 || ne<0 )  
    {  
        printf("INVALID INPUT");  
    }  
    else if(ni==0 && ne==0)  
    {  
        printf("Total estimated Cost : 0.0");  
    }  
    else  
    {  
        for(i=0;i<ni;i++)  
        {  
            scanf("%f",&temp);  
            cost+= int\_p\*temp;  
        }  
        for(i=0;i<ne;i++)  
        {  
            scanf("%f",&temp);  
            cost+= ext\_p\*temp;  
        }  
        printf("Total estimated Cost : %.1f",cost);  
    }  
    return 0;  
}

There are total n number of Monkeys sitting on the branches of a huge Tree. As travelers offer Bananas and Peanuts, the Monkeys jump down the Tree. If every Monkey can eat k Bananas and j Peanuts. If total m number of Bananas and p number of Peanuts are offered by travelers, calculate how many Monkeys remain on the Tree after some of them jumped down to eat.  
At a time one Monkeys gets down and finishes eating and go to the other side of the road. The Monkey who climbed down does not climb up again after eating until the other Monkeys finish eating.  
Monkey can either eat k Bananas or j Peanuts. If for last Monkey there are less than k Bananas left on the ground or less than j Peanuts left on the ground, only that Monkey can eat Bananas(<k) along with the Peanuts(<j).  
Write code to take inputs as n, m, p, k, j and return  the number of Monkeys left on the Tree.  
    Where, n= Total no of Monkeys  
        k= Number of eatable Bananas by Single Monkey (Monkey that jumped down last may get less than k Bananas)  
        j = Number of eatable Peanuts by single Monkey(Monkey that jumped down last may get less than j Peanuts)  
        m = Total number of Bananas  
        p  = Total number of Peanuts  
Remember that the Monkeys always eat Bananas and Peanuts, so there is no possibility of k and j having a value zero

**Example 1:**  
Input Values      
20  
2  
3  
12  
12

**Output Values**  
Number of  Monkeys left on the tree:10  
 **Note:** Kindly follow  the order of inputs as n,k,j,m,p as given in the above example. And output must include  the same format  as in above example(Number of Monkeys left on the Tree:)  
For any wrong input display INVALID INPUT

#include <stdio.h>  
int main()  
{  
    int n,k,j,m,p;  
    float atebanana=0.0,atepeanut=0.0;  
    scanf("%d %d %d %d %d",&n,&k,&j,&m,&p);  
    if(n<0 || k<0 || j<0 || m<0 || p<0)  
    {  
        printf("INVALID INPUT");  
    }  
    else  
    {  
        if(k>0)  
            atebanana =(float)m/k;  
        if(j>0)  
            atepeanut =(float) p/j;  
        n=n-atebanana-atepeanut;  
        printf("Number of Monkeys left on the Tree:%d",n);  
    }  
    return 0;  
}

**FULLY AUTOMATIC VENDING MACHINE –**dispenses your cuppa on just press of button. A vending machine can serve range of products as follows:

Coffee

1. Espresso Coffee
2. Cappuccino Coffee
3. Latte Coffee

Tea

1. Plain Tea
2. Assam Tea
3. Ginger Tea
4. Cardamom Tea
5. Masala Tea
6. Lemon Tea
7. Green Tea
8. Organic Darjeeling Tea

Soups

1. Hot and Sour Soup
2. Veg Corn Soup
3. Tomato Soup
4. Spicy Tomato Soup

Beverages

1. Hot Chocolate Drink
2. Badam Drink
3. Badam-Pista Drink

**Write a program to take input for main menu & sub menu and display the name of sub menu selected in the following format (enter the first letter to select main menu):**

**Welcome to CCD**

**Enjoy your**

**Example 1:**

* Input:
  + c
  + 1
* Output
  + Welcome to CCD!
  + Enjoy your Espresso Coffee!

**Example 2:**

* Input
  + t
  + 9
* **Output**
  + INVALID OUTPUT!

#include<stdio.h>

int main()

{

int submenu=0;

char mainmenu='\0';

printf("Welcome to CCD!");

printf("\n[C]offee\n[T]ea\n[S]oups\n[B]everages\n");

scanf("%c",&mainmenu);

switch(mainmenu)

{

case 'C':

case 'c':printf("1.\tEspresso Coffee\n2.\tCappuccino Coffee\n3.\tLatte Coffee\n");

scanf("%d",&submenu);

switch(submenu)

{

case 1:printf("\no\tEnjoy your Espresso Coffee!");break;

case 2:printf("\no\tEnjoy your Cappuccino Coffee!");break;

case 3:printf("\no\tEnjoy your Latte Coffee!");break;

default:printf("\nINVALID OUTPUT!");

}

break;

case 'T':

case 't':printf("1.\tPlain Tea\n2.\tAssam Tea\n3.\tGinger Tea\n4.\tCardamom Tea\n5.\tMasala Tea\n6.\tLemon Tea\n7.\tGreen Tea\n8.\tOrganic Darjeeling Tea\n");

scanf("%d",&submenu);

switch(submenu)

{

case 1:printf("\no\tEnjoy your Plain Tea!");break;

case 2:printf("\no\tEnjoy your Assam Tea!");break;

case 3:printf("\no\tEnjoy your LGinger Tea!");break;

case 4:printf("\no\tEnjoy your Cardamom Tea!");break;

case 5:printf("\no\tEnjoy your Masala Tea!");break;

case 6:printf("\no\tEnjoy your Lemon Tea\n!");break;

case 7:printf("\no\tEnjoy your Green Tea!");break;

case 8:printf("\no\tEnjoy your Organic Darjeeling Tea!");break;

default:printf("INVALID OUTPUT!");

}

break;

case 'S':

case 's':printf("1.\tHot and Sour Soup\n2.\tVeg Corn Soup\n3.\tTomato Soup\n4.\tSpicy Tomato Soup\n");

scanf("%d",&submenu);

switch(submenu)

{

case 1:printf("\no\tEnjoy your Hot and Sour Soup!");break;

case 2:printf("\no\tEnjoy your Veg Corn Soup!");break;

case 3:printf("\no\tEnjoy your Tomato Soup!");break;

case 4:printf("\no\tEnjoy your Spicy Tomato Soup!");break;

default:printf("\noINVALID OUTPUT!");

}

break;

case 'B':

case 'b':printf("1.\tHot Chocolate Drink\n2.\tBadam Drink\n3.\tBadam-Pista Drink\n");

scanf("%d",&submenu);

switch(submenu)

{

case 1:printf("\no\tEnjoy your Hot Chocolate Drink!");break;

case 2:printf("\no\tEnjoy your Badam Drink!");break;

case 3:printf("\no\tEnjoy your Badam-Pista Drink!");break;

default:printf("\nINVALID OUTPUT!");

}

break;

default:printf("\nINVALID OUTPUT!");

}

return 0;

}

**ALT-TAB Window**

While using a computer, a user uses the ALT-TAB key to switch between  
applications. The ALT-TAB window works on the principle of holding the ALT  
key for MRU (Most Recently Used) listing. Hence, the applications arrange  
themselves in such a way that the most recently used application will be the first  
item in the ALT-TAB window and so forth.  
You are given the list of opened applications and the number of times that the  
user presses the 'Tab key' to switch between applications. Find the final  
arrangement of applications in the ALT-TAB window.  
Example  
in the given picture, Libraries application is focused, which means that holding

#include <stdio.h>

int main()

{

int ar[10],k=0,j=0,i,l=0,ar2[10],n;

printf("number of recent applications");

scanf("%d",&n);

printf("enter the applications\n");

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

printf("number of times tab key pressed");

scanf("%d",&k);

for(j=k,i=1;j>1;j--,i=(i<n-1)?i+1:0);

ar2[0]=ar[i];

for(j=1,l=0;j<n;j=(l!=i)?j+1:j,l++)

{

if(l==i)

continue;

ar2[j]=ar[l];

}

for(i=0;i<n;i++)printf("%d ",ar2[i]);

return 0;

}

### **Question 1: Rat Count House**

**Problem Description :**

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:**

* Return -1 if the array is null
* Return 0 if the total amount of food from all houses is not sufficient for all the rats.
* 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, amount of food in 1st 4 houses is sufficient for all the rats. Thus, output is 4.

#include<stdio.h>

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

{

int totalFoodRequired = r \* unit;

int foodTillNow = 0;

int house = 0;

if (n == 0)

return -1;

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=0,unit=0,n=0,arr[100];

scanf("%d",&r);

scanf("%d",&unit);

scanf("%d",&n);

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

scanf("%d",&arr[i]);

printf("%d",calculate (r, unit, arr, n));

return 0;

}

Question 2:

**(Asked in Accenture OnCampus 10 Aug 2021, Slot 2)**

**Problem Description :**

The Binary number system only uses two digits, 0 and 1 and number system can be called binary string. You are required to implement the following function:

int OperationsBinaryString(char\* str);

The function accepts a string str as its argument. The string str consists of binary digits separated with an alphabet as follows:

* – A denotes AND operation
* – B denotes OR operation
* – C denotes XOR Operation

You are required to calculate the result of the string str, scanning the string to right taking one opearation at a time, and return the same.

**Note:**

* No order of priorities of operations is required
* Length of str is odd
* If str is NULL or None (in case of ), return -1

**Input:**

str: 1C0C1C1A0B1

**Output:**

1

**Explanation:**

The alphabets in str when expanded becomes “1 XOR 0 XOR 1 XOR 1 AND 0 OR 1”, result of the expression becomes 1, hence 1 is returned.

**Sample Input:**

0C1A1B1C1C1B0A0

**Output:**

0

#include<stdio.h>

#include<string.h>

int OperationsBinaryString(char\* str)

{

int i=1;

int a=\*str-'0';

if(str==NULL)

return -1;

str++;

while(\*str!='\0')

{

char p=\*str;

str++;

if(p=='A')

a&=(\*str -'0');

else if (p=='B')

a|=(\*str -'0');

else a^=(\*str -'0');

str++;

}

return a;

}

int main()

{

int len=0;

char s[100];

scanf("%[^\n]s",s);

len=strlen(s);

printf("%d ",OperationsBinaryString(s));

return 0;

}

Question 3: Password Checker

**(Asked in Accenture OnCampus 10 Aug 2021, Slot 3)**

You are given a function.  
int CheckPassword(char str[], int n);  
The function accepts string str of size n as an argument. Implement the function which returns 1 if given string str is valid password else 0.  
str is a valid password if it satisfies the below conditions.

* – At least 4 characters
* – At least one numeric digit
* – At Least one Capital Letter
* – Must not have space or slash (/)
* – Starting character must not be a number

**Assumption:**  
Input string will not be empty.

**Example:**

**Input:**  
aA1\_67  
**Output:**  
1

**Sample Input:**  
a987 abC012  
**Output:**  
0

#include<stdio.h>

#include<string.h>

int CheckPassword(char str[],int n)

{

int a=0,cap=0,nu=0;

//At least 4 characters

if(n<4) return 0;

//Starting character must not be a number

if(str[0]-'0'>=0 && str[0]-'0'<=9) return 0;

while(a<n)

{

//Must not have space or slash (/)

if(str[a]==' ' || str[a]=='/') return 0;

//counting capital letters

if(str[a]>=65&&str[a]<=90) {cap++;}

//counting numeric digit

else if(str[a]-'0'>=0 && str[a]-'0'<=9) nu++;

//incrementing for while loop

a++;

}

// returns 1 if there are > 0 numeric digits and capital letters

return cap>0 && nu>0 ;

}

int main()

{

int len=0;

char s[100];

scanf("%[^\n]s",s);

len=strlen(s);

printf("%d",CheckPassword(s,len));

return 0;

}

Question 4:

**(Asked in Accenture OnCampus 11 Aug 2021, Slot 1)**

**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:**

* n=6
* 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.

#include<stdio.h>

#include<stdlib.h>

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

int count = 0,i=0;

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

{

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

{

count++;

}

}

return count > 0 ? count : -1;

}

int main() {

int n=0, \*arr,i=0,num=0,diff=0;

scanf("%d",&n);

arr=(int\*)malloc(sizeof(int)\*n);

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

scanf("%d",&arr[i]);

}

scanf("%d",&num);

scanf("%d",&diff);

printf("%d",findCount(n, arr, num, diff));

return 0;

}

return 0;

}

Question 5 :

**(Asked in Accenture OnCampus 11 Aug 2021, Slot 2)**

**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

#include<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;  
 }   
 }  
 if(sum2>sum1)  
 return sum2 - sum1;  
 else   
 return sum1-sum2;  
}  
  
int main()  
{  
 int n=0, m=0;  
 scanf("%d",&n);  
 scanf("%d",&m);  
 printf("%d", differenceofSum(n, m));  
 return 0;  
}

Question:6

**(Asked in Accenture OnCampus 11 Aug 2021, Slot 3)**

**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  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 as even

**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 smallest 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

#include <stdio.h>;

int largeSmallSum(int \*array, int n)

{

int answer, i, j, temp;;

int even[n], odd[n];

int evenount = 0, oddount = 0;

if(n<=3)

{

answer = 0;

}

else

{

even[0] = array[0];

evenount = 1;

for(i=1; i<n; i++) //reating two array even and odd

{

if(i%2==0)

{

even[evenount] = array[i];

evenount++;

}

else

{

odd[oddount] = array[i];

oddount++;

}

}

for(i=0; i<evenount; i++) //sorting of even array

{

for(j=i+1; j<evenount; j++)

{

if(even[i]>even[j])

{

temp = even[i];

even[i] = even[j];

even[j] = temp;

}

}

}

for(i=0; i<oddount; i++) //sorting of odd array

{

for(j=i+1; j<oddount; j++)

{

if(odd[i]>odd[j])

{

temp = odd[i];

odd[i] = odd[j];

odd[j] = temp;

}

}

}

answer = even[evenount-2] + odd[1];

}

return answer;

}

int main()

{

int n, result, i;

sanf("%d",&n);

int array[n];

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

{

sanf("%d",&array[i]);

}

result = largeSmallSum(array, n);

printf("%d",result);

return 0;

}

Question:7

**(Asked in Accenture On-campus 12 Aug 2021, Slot 1)**

**Implement the following Funtion**

def ProdutSmallestPair(sum, arr)

The funtion aepts an integers sum and an integer array arr of size n. Implement the funtion to find the pair, (arr[j], arr[k]) where j!=k, Suh 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 sum 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, Produt of (2, 1) 2\*1 = 2. Thus, output is 2

**Sample Input**

sum:4

n: 6

Arr:9 8 3 -7 3 9

**Sample Output**

-21

#include<stdio.h>

int produtSmallestPair(int \*array, int n, int sum)

{

int answer, temp, i, j, hek;

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;

}

}

}

hek = array[0] + array[1];

if(hek<=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 = produtSmallestPair(array, n, sum);

printf("%d",result);

return 0;

}

**(Asked in Accenture Off campus 1 Aug 2021, Slot 1)**

**Implement the following funtions.a**

har\*MoveHyphen(har str[],int n);

The funtion aepts a string “str” of length ‘n’, that ontains alphabets and hyphens (-). Implement the funtion to move all hyphens(-) in the string to the front of the given string.

**NOTE**:- Return null if str is null.

**Example :-**

* **Input**:
  + str.Move-Hyphens-to-Front
* **Output:**
  + —MoveHyphenstoFront

**Explanation:-**

The string “Move-Hyphens -to-front” has 3 hyphens (-), whih are moved to the front of the string, this output is “— MoveHyphen”

**Sample Input**

* Str: String-ompare

**Sample Output-**

* -Stringompare

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

char\* MoveHyphen (char\* s, int n)

{

int count =0,i;

char \*res=(char\*)malloc(sizeof(char)\*n);

for(i=0; i<n; )

{

if(s[i]=='-')

{

int j;

count ++;

for(j=i;s[j]!='\0';s[j]=s[j+1],j++);//repacing the character

s[j]='\0';

}

else i++;

}

while(count--){

strcat(res,"-" );

}

strcat(res,s );

return res;

}

int main ()

{

char s[100];

int n=0;

scanf("%s",s);

n= strlen(s);

printf("%s",MoveHyphen(s, n));

return 0;

}

**Question:10**

**(Asked in Accenture Off-campus 1 Aug 2021, Slot 2)**

### **Question:10**

**(Asked in Accenture Offcampus 1 Aug 2021, Slot 2)**

**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 accept 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

#include<stdio.h>  
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;  
 }  
 while(num1!=0)  
 {  
 p=num1%10;  
 sum=carry+p;  
 if(sum>9)  
 {  
 carry=1;  
 count++;  
 }  
 else   
 carry=0;  
 num1=num1/10;  
 }  
 while(num2!=0)  
 {  
 q=num2%10;  
 sum=carry+q;  
 if(sum>9)  
 {  
 carry=1;  
 count++;  
 }  
 else   
 carry=0;  
 num2=num2/10;  
 }  
 return count;  
}  
  
int main()  
{  
 int x, y, a;  
 scanf("%d",&x);  
 scanf("%d",&y);  
 a = numberOfCarries(x, y);  
 printf("%d",a);  
   
 return 0;  
}

### **Question:11**

**(Asked in Accenture Offcampus 1 Aug 2021, Slot 3)**

**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: apples
  + ch1:a
  + ch2:p
* **Output:**
  + paales

**Explanation:**

‘A’ in original string is replaced with ‘p’ and ‘p’ in original string is replaced with ‘a’, thus output is paales.

#include <stdio.h>  
#include <string.h>  
  
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;  
 }  
 }  
 printf("%s",str);  
}  
  
int main()  
{  
 char a[100];  
 char b, c;  
 int len;  
 scanf("%s",a);  
 scanf("%s",&b);  
 scanf("%s",&c);  
  
 len = strlen(a);  
 ReplaceCharacter(a, len, b, c);  
  
 return 0;  
}

### **Question:12**

**(Asked in Accenture Off campus 2 Aug 2021, Slot 1)**

**Problem Statement**

**You are required to implement the following function.**

Int OperationChoices(int c, int n, 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

#include<stdio.h>  
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;  
 scanf("%d",&x);  
 scanf("%d",&y);  
 scanf("%d",&z);  
  
 result = operationChoices(x, y, z);  
  
 printf("%d",result);  
 return 0;  
}

### **Question:13**

**(Asked in Accenture Offcampus 2 Aug 2021, Slot 2)**

**Problem Statement**

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 so faqrd in a variable , 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.

#include<bits/std.h>  
using namespace std;  
  
int count(int n){  
   
 int c = 0;  
   
 while (n % 2 == 0 && n != 0){  
   
 c++;  
 n = n / 2;  
   
 }  
return c;  
   
}  
  
int maxExponents(int a, int b){  
   
 int max = 0, num = 0, ans;  
   
 for (int i = a; i <= b; i++){  
   
 int temp = count(i);  
   
 if (temp > max){  
   
 max = temp;  
 num = i;  
 }  
   
 }  
   
 return num;  
}   
  
int main ()   
{  
 int a, b;  
 cin >> a>>b;  
   
 cout<<maxExponents(a, b);  
   
 return 0;  
}

### **Question : 14**

**(Asked in Accenture Off campus 2 Aug 2021, Slot 3)**

**Problem Statement**

**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**  
510

## 

#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;  
}

### **Question 15**

**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

7

#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   
}

### **Question: 16**

**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.

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

**Test Cases**:

**Test Case 1:**  
**Input:**  
5  
**Expected Result Value:**  
5, 10, 15, 20, 25, 30, 35, 40, 45, 50  
275

**Test Case 2:**  
**Input:**  
12  
**Expected Result Value:**  
12, 24, 36, 48, 60, 72, 84, 96, 108, 120  
660

#include <stdio.h>  
int main()  
{  
 int n, i, value=0, sum=0;   
 printf("Enter the number for which you want to know the table : ");  
 scanf("%d",&n);  
  
 for(i=1; i<=10; ++i)  
 {  
 value = n \* i;  
 printf("%d ",value);  
 sum=sum+value;  
 }  
  
 printf("\nsum is %d",sum);  
 return 0;  
}

### **Question : 17**

**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 palindrome numbers.

**Test Cases:**

**Test Case 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.

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

**Question : 18**

**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

Output: 10.783510

Distance is calculated as : **sqrt(x2-x1)2 + (y2-y1)2**

#include<stdio.h>  
#include<math.h>  
int main()  
{  
 float x1,y1,x2,y2,x3,y3;  
 printf("Enter x1,y1 : ");  
 scanf("%f %f",&x1,&y1);  
 printf("Enter x2,y2 : ");  
 scanf("%f %f",&x2,&y2);  
 printf("Enter x3,y3 : ");  
 scanf("%f %f",&x3,&y3);  
   
 float firstDiff =(float) sqrt (pow (x2 - x1, 2) + pow (y2 - y1, 2));  
 float secondDiff =(float) sqrt (pow (x3 - x2, 2) + pow (y3 - y2, 2));  
 float thirdDiff =(float) sqrt (pow (x3 - x1, 2) + pow (y3 - y1, 2));  
 printf("%f",(firstDiff + secondDiff + thirdDiff));  
  
 return 0;   
}

**Question : 19**

**Find the maximum value and its index in the array**

**Problem Statement :**

You are given a function, void MaxInArray(int arr[], int length); The function accepts an integer array ‘arr’ of size ‘length’ as its argument. Implement the function to find the maximum element of the array and print the maximum element and its index to the standard output

(STDOUT). The maximum element and its index should be printed in separate lines.

Note:

* Array index starts with 0
* Maximum element and its index should be separated by a line in the output
* Assume there is only 1 maximum element in the array
* Print exactly what is asked, do not print any additional greeting messages

**Example:**

Input:

23 45 82 27 66 12 78 13 71 86

Output:

86

9

Explanation:

86 is the maximum element of the array at index 9.

#include<stdio.h>

#define INT\_MIN -2147483647 - 1

void MaxInArray (int arr[], int length)

{

int max = INT\_MIN, index = -1;

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

{

if (arr[i] > max){

max = arr[i];

index = i;

}

}

printf("%d\n%d",max,index);

}

int main ()

{

int n,arr[100], i = 0;

scanf("%d",&n);

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

scanf("%d",&arr[i]);

MaxInArray (arr, n);

return 0;

}

**Question : 20**

**Autobiographical Number**

**Problem Statement :**

An Autobiographical Number is a number N such that the first digit of N represents the count of how many zeroes are there in N, the second digit represents the count of how many ones are there in N and so on.

You are given a function,**def FindAutoCount(n):**

The function accepts string “n” which is a number and checks whether the number is an autobiographical number or not. If it is, an integer is returned, i.e. the count of distinct numbers in ‘n’. If not, it returns 0.

**Assumption:**

* The input string will not be longer than 10 characters.
* Input string will consist of numeric characters.

**Note:**

If string is None return 0.

**Example:**

Input:

n: “1210”

Output:

3

Explanation:

0th position in the input contains the number of 0 present in input, i.e. 1, in 1st position the count of number of 1s in input i.e. 2, in 2nd position the count of 2s in input i.e. 1, and in 3rd position the count of 3s i.e. 0, so the number is an autobiographical number.

Now unique numbers in the input are 0, 1, 2, so the count of unique numbers is 3. So 3 is returned.

#include<bits/std.h>  
using namespace std;  
  
int FinndAutoCount (string n)   
{  
 int sum=0;  
 set < char >st;  
   
 for (int i = 0; i < n.size (); i++)  
 {   
 sum += (n[i]-'0');  
 st.insert (n[i]);  
 }  
 if(sum != n.size())  
 return 0;  
   
 return st.size ();  
}  
  
int main ()   
{  
 string n;  
 cin >> n;  
   
 cout << FinndAutoCount (n);  
   
 return 0;  
}

**Count the number of co-prime pairs in an array. (**Any two numbers whose GCD is 1 are be called as co-prime)

**Input:**

The first line contains an integer T, total number of elements. Then follow T elements.

**Output:**

Count the number of co-prime pairs in an array.

**Constraints:**

1 ? T ? 25

1 ? elements ? 100

**Sample Input and Output:**

**Input 1:**

3

1 2 3

**Output 1:**

3

Here, Co-prime pairs are (1, 2), (2, 3), (1, 3)

**Input 2:**

4

4 8 3 9

**Output 2:**

4

Here, Co-prime pairs are (4, 3), (8, 3), (4, 9 ), (8, 9 )

#include<stdio.h>

int coprime(int a, int b)

{

int gcd;

while ( a != 0 )

{

gcd = a; a = b%a; b = gcd;

}

if(gcd == 1)

return 1;

elsereturn 0;

}

int count\_pairs(int arr[], int n)

{

int count = 0;

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

{

for (int j = i + 1; j < n; j++)

{

if (coprime(arr[i], arr[j]))

count++;

}

}

return count;

}

int main()

{

int n;

scanf("%d", &n);

int a[25], i;

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

scanf("%d", &a[i]);

printf("%d", count\_pairs(a, n));

return 0;

}

**Search for Nth Occurrence**

Given an array, number to search (say e1), and occurrence (say n), print the index of the nth occurrence of e1 in the array. If e1 does not occur n times, then print the index as -1.

**Input and Output:**

Get the size of an array and get elements one by one. Input the number to be searched and occurrence. For example, 7 => Size of an array 1 4 6 7 6 3 6 => array elements 6 => number to be searched 3 => 3rd occurrence of number 6 Output: 6 Explanation: Number 6, 3rd occurrence position is 6

**Sample Input and Output:**

**Input:**

7

1 4 6 7 6 3 6

6

3

**Output:**

6

#include<stdio.h>

int main()

{

int a[100],n,i,e1,size,count=0;

scanf("%d",&size);

for(i=0;i<size;i++)

scanf("%d",&a[i]);

scanf("%d",&e1);

scanf("%d",&n);

for(i=0;i<size;i++)

{

if(e1==a[i])

count++;

//If 'n'th occurrence found then print it's index and exit.

if(count==n)

{

printf("%d",i);

return 0;

}

}

//If 'n' occurrence not found then print '-1'.

printf("%d",-1);

return 0;

}

**Search for an element in an array:**

Program to search for an element in the given array.

**Input and Output:**

The input consists of n + 2 lines. The first line consists a single integer n, The next n lines consist of 1 integer element part of the array. The last line consists of an integer to be searched. Output found or missing based on whether the element is present in the array or not. Note: max value of n is 100.

**Sample Input and Output:**

**Input 1:**

**3**

1 2 3

6

**Output 1:**Missing

**Input 2:**

**3**

1 2 3

2

**Output 2:**Found

#include<stdio.h>#define MAX\_SIZE 20

int main()

{

int n, i, j, min\_index, array[MAX\_SIZE], x;

scanf("%d", &n);

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

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

scanf("%d", &x);

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

{

if(x == array[i])

{

printf("Foundn");

return 0;

}

}

printf("Missingn");

return 0;

}

**Search index in a sorted array:**

Program to find the target value in a two-dimensional matrix.

**Input and Output:**

Get a target element and return its coordinates. If the value didn't exist, the program had to return (-1,-1).The first line of input is the sizeof row and column, followed rxc elements. The third line of input is the element to be searched in the rxc matrix.

**Sample Input and Output:**

**Input 1:**

4 2

0 9 8 7 6 5 4 3

3

**Output 1:**

(3, 1)

#include<stdio.h>

int main()

{

int i, j, count = 0;

int arr[10][10], search, r, c;

scanf("%d %d", &r, &c);

for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

scanf("%d", &arr[i][j]);

}

scanf("%d", &search);

for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

{

if (arr[i][j] == search)

{

printf("(%d , %d)n", i, j);

count++;

}

}

}

if (count == 0)

printf("(-1,-1)");

return 0;

}

Program to find all symmetric pairs in an array is discussed here. Two pairs (p,q) and (r,s) are said to be symmetric when q is equal to r and p is equal to s. For example, (5,10) and (10,5) are symmetric pairs.

For example,

Consider a 2D array,

**Input:**

arr [6] [2] = {{1, 2}, {3, 4}, {5, 6}, {2, 1}, {4, 3},{10,11}}

**Output:**

{1,2} and {2,1} are symmetric

{3,4} abd {4,3} are symmetric

This problem can be solved in two different ways.

**Method 1:** Using two loops, one loop to traverse the pairs and the other loop to check if similar pair is existing.

**Method 2:**An efficient way to solve this problem is to use **hashing**. Insert each array pairs into the hash table with the first element of the pair as the **key** and the second element as the **value**. Traverse the hash table to check if the pairs are found again.

## **Algorithm to find all symmetric pairs in an array**

1. Input the array from the user.
2. Use two loops.
3. One loop for traversing the array and the other loop to check if symmetric pair is found in the array.
4. If symmetric pair is found, print the pairs.

Program to find all symmetric pairs in an array is given below.

#include <stdio.h>

void symmetric\_array\_pair(int arr1[], int arr2[], int m, int n)

{

int i,j;

for(i=0;i<m;i++)

{

for(j=0;j<m;j++)

{

if(arr1[i] == arr2[j] && arr1[j]== arr2[i])

{

printf("(%d,%d)\t",arr1[i],arr1[j]);

continue;

}

}

}

}

int main()

{

int m,n;

scanf("%d%d",&m,&n);

int i,j;

int arr1[m],arr2[n];

for(i=0;i<m;i++)

{

scanf("%d",&arr1[i]);

}

for(i=0;i<m;i++)

{

scanf("%d",&arr2[i]);

}

symmetric\_array\_pair(arr1,arr2,m,n);

return 0;

}