PROBLEM SOLVING

(Solving Various Problems Using C Language)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

D Vidya Reddy

221710307020

Under the Guidance of

Assistant Professor

Department Of Computer Science Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

DECLARATION

I submit this industrial training work entitled **“SOLVING VARIOUS PROBLEMS USING C LANGUAGE**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science Engineering**”. I declare that it was carried out independently by me under the guidance of *************, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD D Vidya Reddy
Date: 20-07-2020 221710307020

GITAM (DEEMED TO BE UNIVERSITY)
Hyderabad-502329, India
Dated : 20-07-2020

CERTIFICATE

This is to certify that the Industrial Training Report entitled **“SOLVING VARIOUS PROBLEMS USING C LANGUAGE”** is being submitted by D. Vidya Reddy (221710307020) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20.

It is faithful record work carried out by her at the **Computer Science Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

****
Assistant Professor
Department of CSE

Dr. S. Phani Kumar
Assistant Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. N. Seetharamaiah,** Principal, GITAM Hyderabad

I would like to thank respected **Mr.S Phani Kumar,** Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties ******* who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

D. Vidya Reddy
221710307020

Table of Contents

1 Introduction to the project

Problem Solving is the Process of Designing and carrying out certain steps to reach a Solution. There are eight problems which are listed below are of different complexity and require different approach and logics in order to achieve desired Output/Solution..

1. **Game of Primes :** In this problem we have to find the number which is both happy and prime.
2. **Television Sets :** In this Problem we find the number of TV rooms and non-TV needed for a hospital to meet its revenue.
3. **Marathon Winner :** In this problem we find the winner of a race which is organized by time.
4. **Minimum Product Array :** In this problem we find the minimum sum of product of 2 arrays by changing the values of 1 particular array by a given number of times.
5. **Holes and Balls :** In this problem we find the balls which will fit in the holes according to their size.
6. **Balancing Stars :** In this problem we find the number of balanced brackets and if there are 2 or more stars in between every balanced brackets.
7. **Grooving Monkeys :** In this problem we find the time taken by monkeys to come to their initial position where they are following a particular pattern while dancing in a party.
8. **Consecutive Prime Sum :** In this problem we find the count of prime numbers which can be expressed as a sum of other consecutive prime numbers.

I have executed projects in C language. I have used DEV C++ to execute the codes .

2 Problem 1
Game of Primes

In this problem we have to find the number which is both happy and prime.

2.1 Problem Statement:-

In a global Mathematics contest, the contestants are told to invent some special numbers which can be built by adding the squares of its digits. Doing this perpetually, the numbers will end up to 1 or 4. In this Problem we find the number of TV rooms and non-TV needed for a hospital to meet it's revenue.

If a positive integer ends with 1, then it is called the Number of Game.

An example from above is:

13 = 1^2 + 3^2 = 1+9 = 10 (Step:1)

10 = 1^2 + 0^2 = 1+0 = 1 (Step:2), iteration ends in Step 2 since number ends with 1

Then in next round, the contestants are asked to combine their newly invented number, i.e. Number of Game with prime numbers.

Now, being a smart programmer, write a program to help the contestants to find out the Nth combined number within any given range, where N can be any integer.

Input Format:

Input consists of 3 integers X, Y, N, one on each line. X and Y are upper and lower limits of the range. The range is inclusive of both X and Y. Find Nth number in range [X,Y].

Line 1	X,where X is the upper limit of the range
Line 2	Y,where Y is the lower limit of the range
Line 3	N,where Nth element of the series is required

Constraints:

X <= Y

X > 0

N > 0

Output Format:

Output will show the Nth element of the combined series lying in the range between X and Y.

Line 1	For Valid Input,print U,where U is the Nth element of the combined number series lying in the range between X and Y. Or No number is present at this index For Invalid Input,print Invalid Input

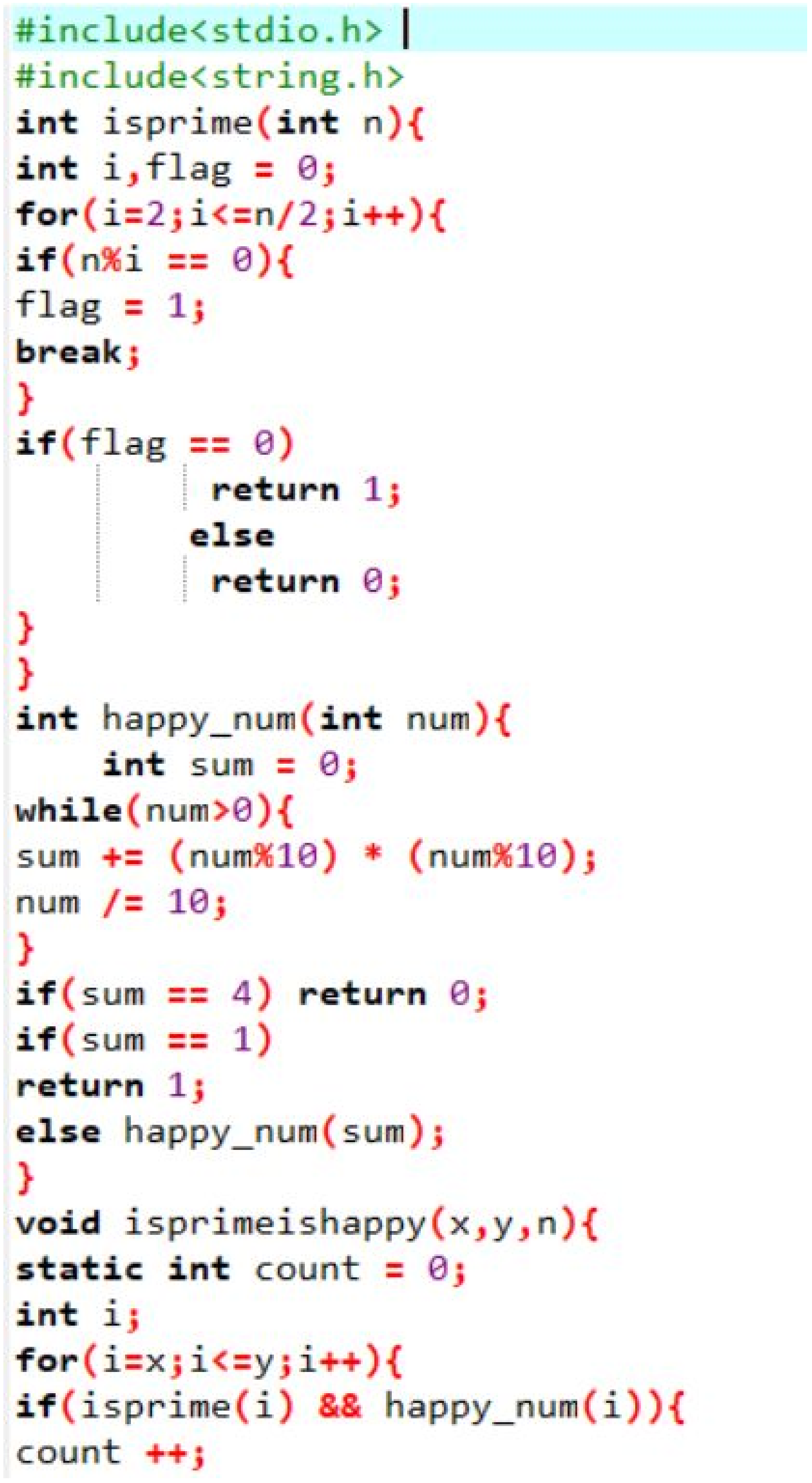
Sample Input and Output

SNo.	Input	Output
1	30 3	9
2	2 33 5	No number is present at this index
3	- 5 @ 4	Invalid Input

Concepts Used To Solve:-

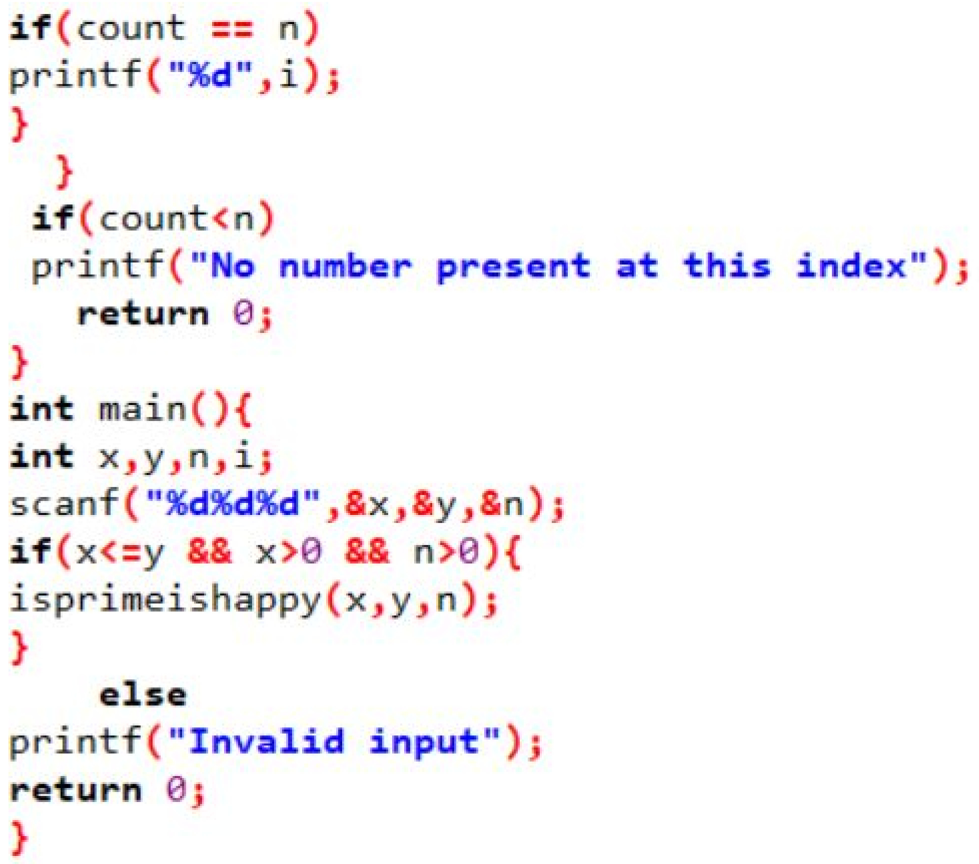
Two functions are used in the problem where one used for loop with if condition to find if a number is prime and another is used for loop with if condition to find if the number is happy and if both the conditions are satisfied then that number will be printed.

2.2 Coding



```c
#include<stdio.h>
#include<string.h>
int isprime(int n){
int i,flag = 0;
for(i=2;i<=n/2;i++){
if(n%i == 0){
flag = 1;
break;
}
if(flag == 0)
         return 1;
        else
         return 0;
}
}
int happy_num(int num){
    int sum = 0;
while(num>0){
sum += (num%10) * (num%10);
num /= 10;
}
if(sum == 4) return 0;
if(sum == 1)
return 1;
else happy_num(sum);
}
void isprimeishappy(x,y,n){
static int count = 0;
int i;
for(i=x;i<=y;i++){
if(isprime(i) && happy_num(i)){
count ++;
```

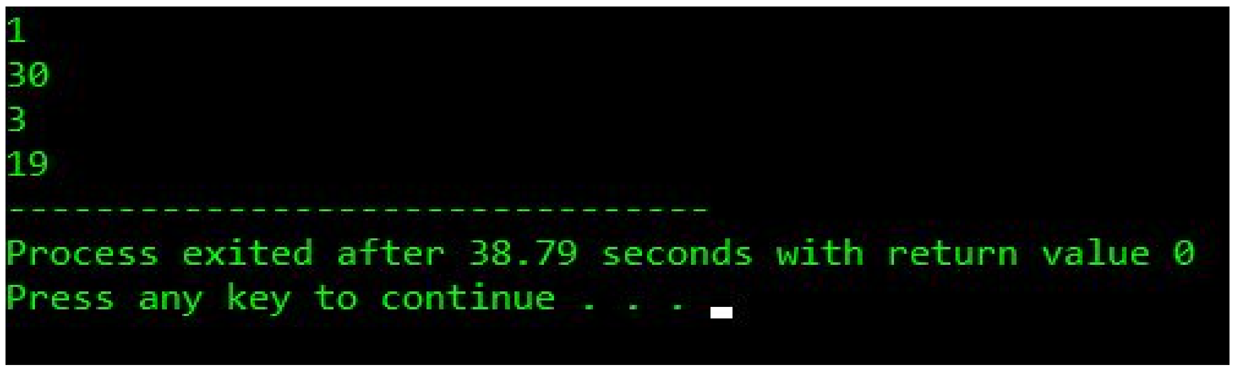
fig 2.2.1



```
if(count == n)
printf("%d",i);
}
  }
 if(count<n)
 printf("No number present at this index");
   return 0;
}
int main(){
int x,y,n,i;
scanf("%d%d%d",&x,&y,&n);
if(x<=y && x>0 && n>0){
isprimeishappy(x,y,n);
}
    else
printf("Invalid input");
return 0;
}
```

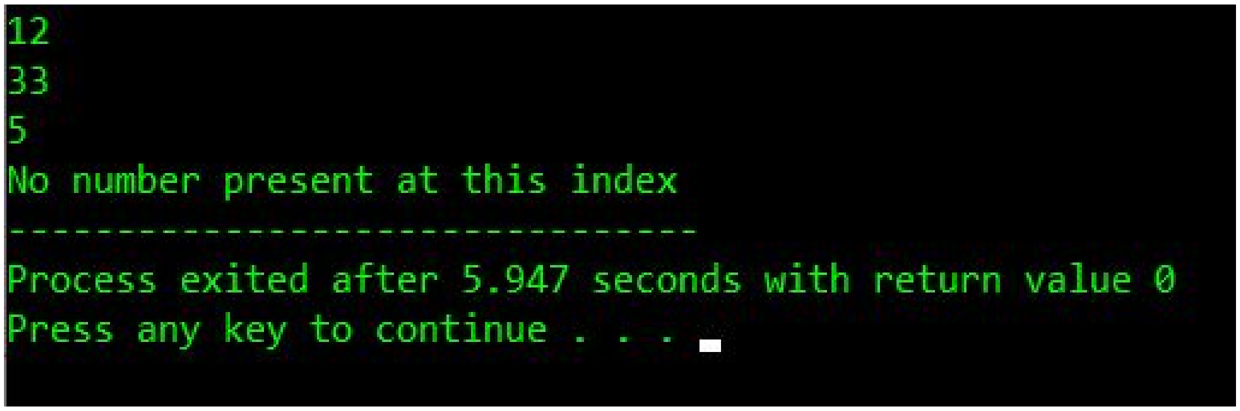
fig 2.2.2

2.3Output



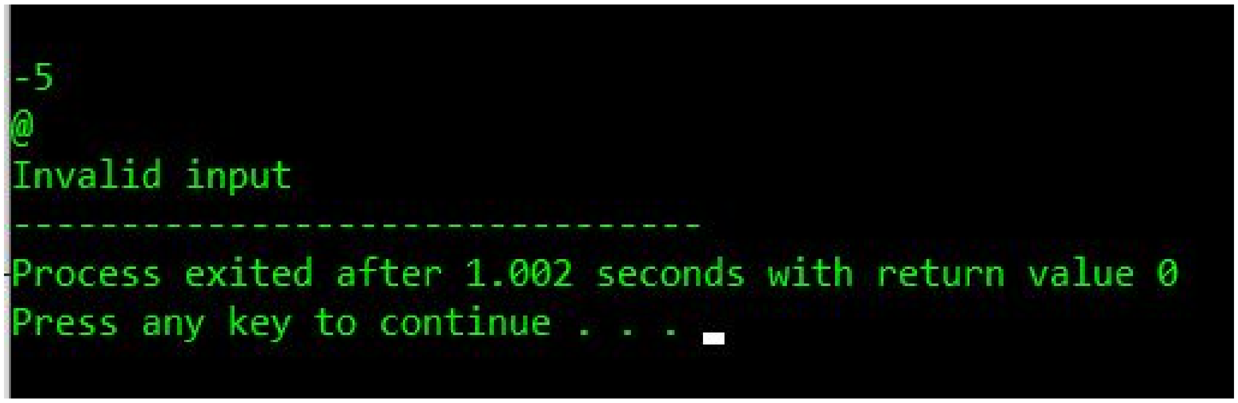
```
1
30
3
19
--------------------------------
Process exited after 38.79 seconds with return value 0
Press any key to continue . . .
```

fig 2.3.1



```
12
33
5
No number present at this index
--------------------------------
Process exited after 5.947 seconds with return value 0
Press any key to continue . . .
```

fig 2.3.2



```
-5
@
Invalid input
--------------------------------
Process exited after 1.002 seconds with return value 0
Press any key to continue . . .
```

fig 2.3.3

3 Problem 2:
Television Sets

In this Problem we find the number of TV rooms and non-TV needed for a hospital to meet its revenue.

3.1 Problem Statement:-

Dr. Vishnu is opening a new world class hospital in a small town designed to be the first preference of the patients in the city. Hospital has N rooms of two types – with TV and without TV, with daily rates of R1 and R2 respectively.
However, from his experience Dr. Vishnu knows that the number of patients is not constant throughout the year, instead it follows a pattern. The number of patients on any given day of the year is given by the following formula –
(6-M)^2 + |D-15| ,
where M is the number of month (1 for jan, 2 for feb …12 for dec) and D is the date (1,2…31).
All patients prefer without TV rooms as they are cheaper, but will opt for with TV rooms only if without TV rooms are not available. Hospital has a revenue target for the first year of operation. Given this target and the values of N, R1 and R2 you need to identify the number of TVs the hospital should buy so that it meets the revenue target. Assume the Hospital opens on 1st Jan and year is a non-leap year.

Constraints

Hospital opens on 1st Jan in an ordinary year
5 <= Number of rooms <= 100
500 <= Room Rates <= 5000
0 <= Target revenue < 90000000

Input Format

First line provides an integer N that denotes the number of rooms in the hospital
Second line provides two space-delimited integers that denote the rates of rooms with TV (R1) and without TV (R2) respectively
Third line provides the revenue target

Output

Minimum number of TVs the hospital needs to buy to meet its revenue target. If it cannot achieve its target, print the total number of rooms in the hospital.

Test Case

Example-1 :

Input

20

1500 1000

7000000

Output

14

Explanation

Using the formula, number of patients on 1st Jan will be 39, on 2nd Jan will be 38 and so on. Considering there are only twenty rooms and rates of both type of rooms are 1500 and 1000 respectively, we will need 14 TV sets to get revenue of 7119500. With 13 TV sets Total revenue will be less than 7000000

Example-2 :

Input

10

1000 1500

10000000

Output

10

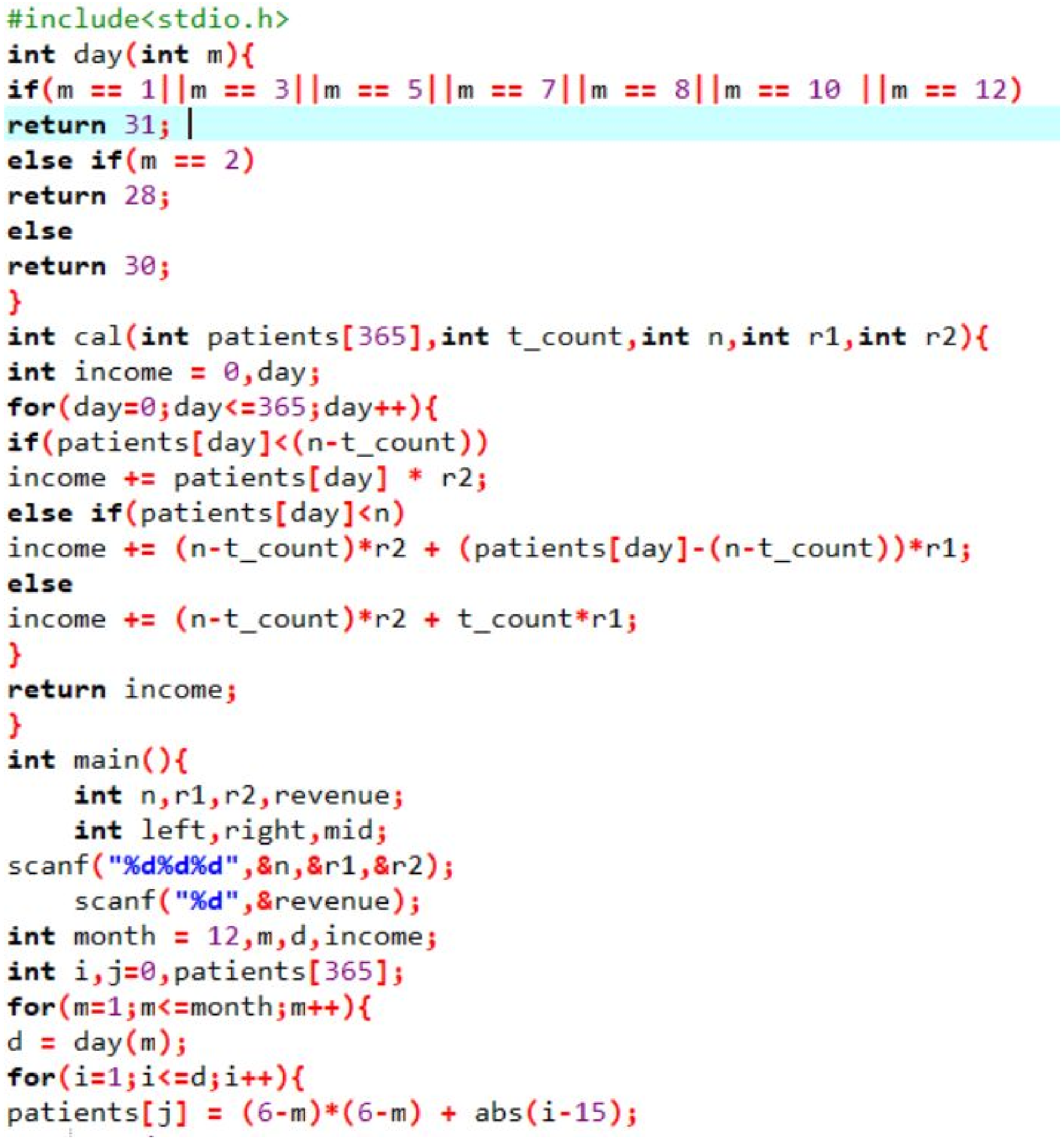
Explanation

In the above example, the target will not be achieved, even by equipping all the rooms with TV. Hence, the answer is 10 i.e. total number of rooms in the hospital.

Concepts Used to solve:-

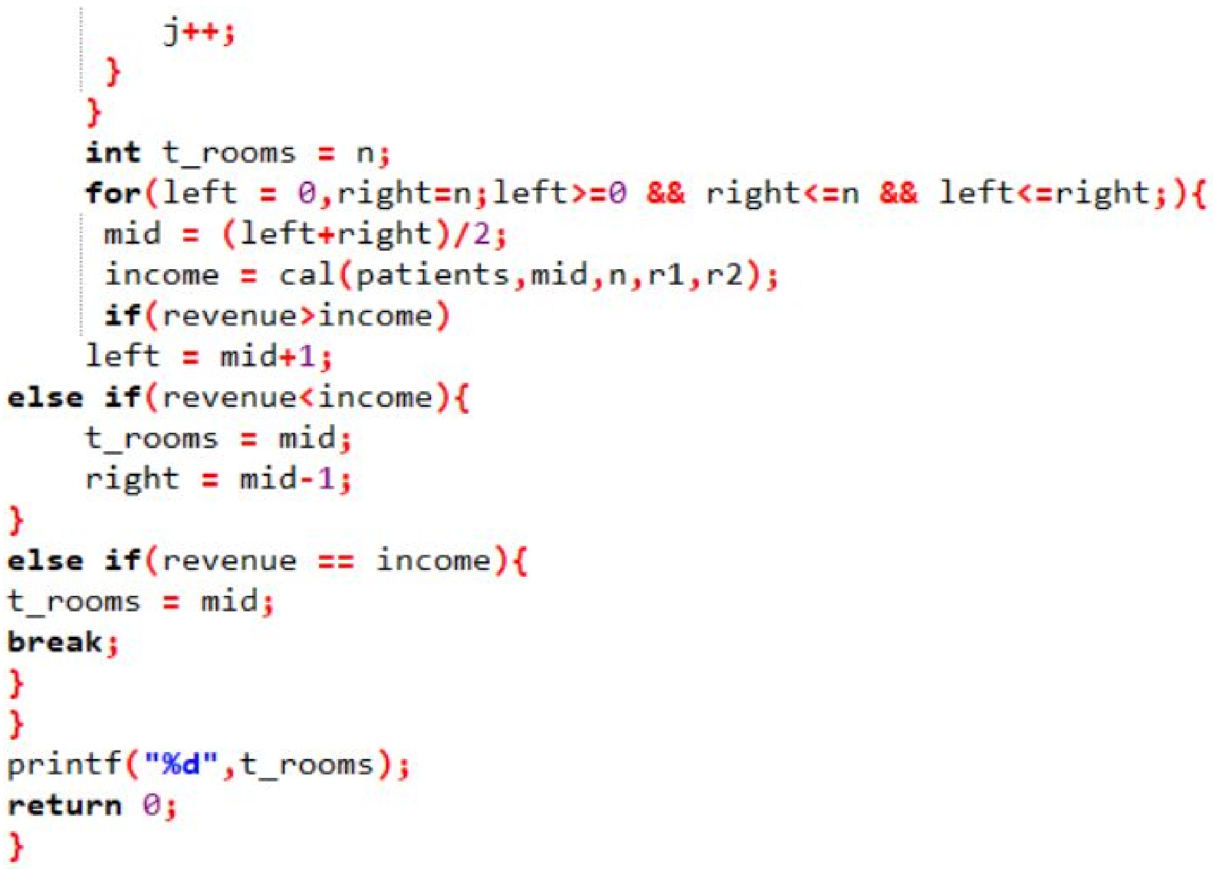
Array is used in the program to hold the count of patients of each day in a year and using functions to calculate income for the year considering the rooms with TV and if the income doesn't match the revenue the increase the TV count else print the number of TV present.

3.2Coding



```
#include<stdio.h>
int day(int m){
if(m == 1||m == 3||m == 5||m == 7||m == 8||m == 10 ||m == 12)
return 31;
else if(m == 2)
return 28;
else
return 30;
}
int cal(int patients[365],int t_count,int n,int r1,int r2){
int income = 0,day;
for(day=0;day<=365;day++){
if(patients[day]<(n-t_count))
income += patients[day] * r2;
else if(patients[day]<n)
income += (n-t_count)*r2 + (patients[day]-(n-t_count))*r1;
else
income += (n-t_count)*r2 + t_count*r1;
}
return income;
}
int main(){
    int n,r1,r2,revenue;
    int left,right,mid;
scanf("%d%d%d",&n,&r1,&r2);
    scanf("%d",&revenue);
int month = 12,m,d,income;
int i,j=0,patients[365];
for(m=1;m<=month;m++){
d = day(m);
for(i=1;i<=d;i++){
patients[j] = (6-m)*(6-m) + abs(i-15);
```

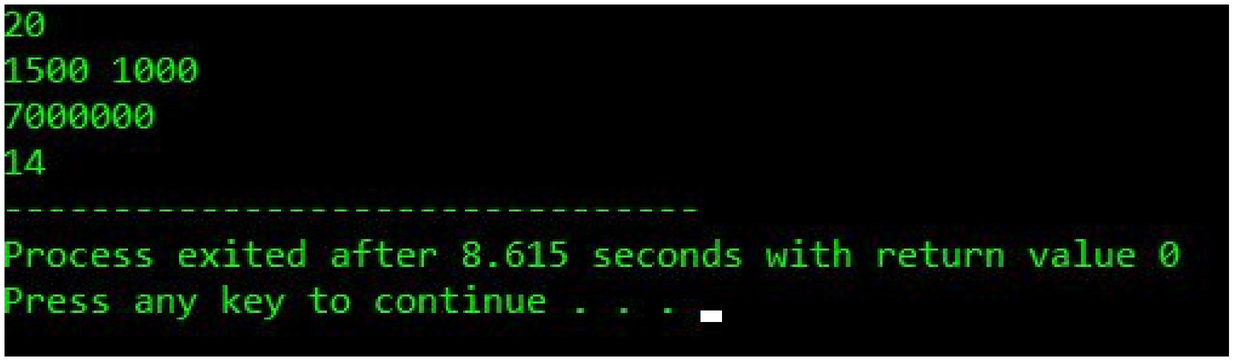
fig 3.2.1



```
        j++;
    }
   }
   int t_rooms = n;
   for(left = 0,right=n;left>=0 && right<=n && left<=right;){
    mid = (left+right)/2;
    income = cal(patients,mid,n,r1,r2);
    if(revenue>income)
   left = mid+1;
else if(revenue<income){
   t_rooms = mid;
   right = mid-1;
}
else if(revenue == income){
t_rooms = mid;
break;
}
}
printf("%d",t_rooms);
return 0;
}
```

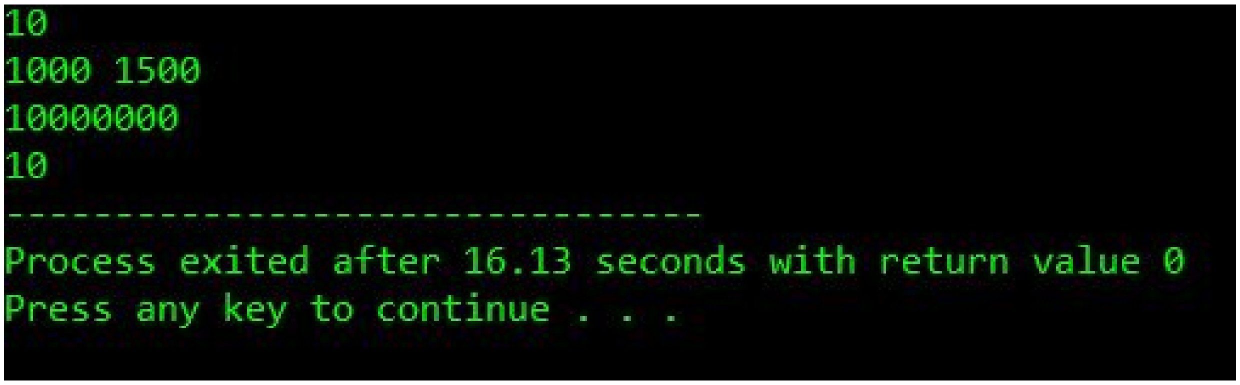
fig 3.2.2

3.3Output



```
20
1500 1000
7000000
14
--------------------------------
Process exited after 8.615 seconds with return value 0
Press any key to continue . . .
```

fig 3.3.1



```
10
1000 1500
10000000
10
--------------------------------
Process exited after 16.13 seconds with return value 0
Press any key to continue . . .
```

fig 3.3.2

4 Problem 3
Marathon Winner

In this problem we find the winner of a race which is organized by time.

4.1 Problem Statement:-

Race is generally organized by distance but this race will be organized by time. In order to predict the winner we will check every 2 seconds.

Let's say total race time is 7 seconds we will check for (7-1) seconds.

For 7 sec : we will check who is leading at 2 sec, 4 sec, and 6 sec.

Participant who is leading more number of times is winner from prediction perspective.

Now our task is to predict a winner in this marathon.

Note :

1. At particular time let say at 4th second, top two (top N, in general) participants are at same distance, then in this case both are leading we'll increase count for both(all N).
2. And after calculating at all time slices, if the number of times someone is leading, is same for two or more participants, then one who come first in input sequence will be the winner.

Ex. If participant 2 and 3 are both leading with same number, participant 2 will be the winner.

Constraints

1 <= T <= 100

1 <= N <= 100

Input Format

First line contains a single integer N denoting the number of participants.

Second line contains a single integer T denoting the total time in seconds of this Marathon.

Next N lines (for each participant) are as follows :

We have T+1 integers seperated by space.

First T integers are as follows :

ith integer denotes number of steps taken by the participant at the i-th second.

T+1st integer denotes the distance (in meters) of each step.

Output

Index of Marathon winner, where index starts with 1.

Test Case

Input

```
3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4
```

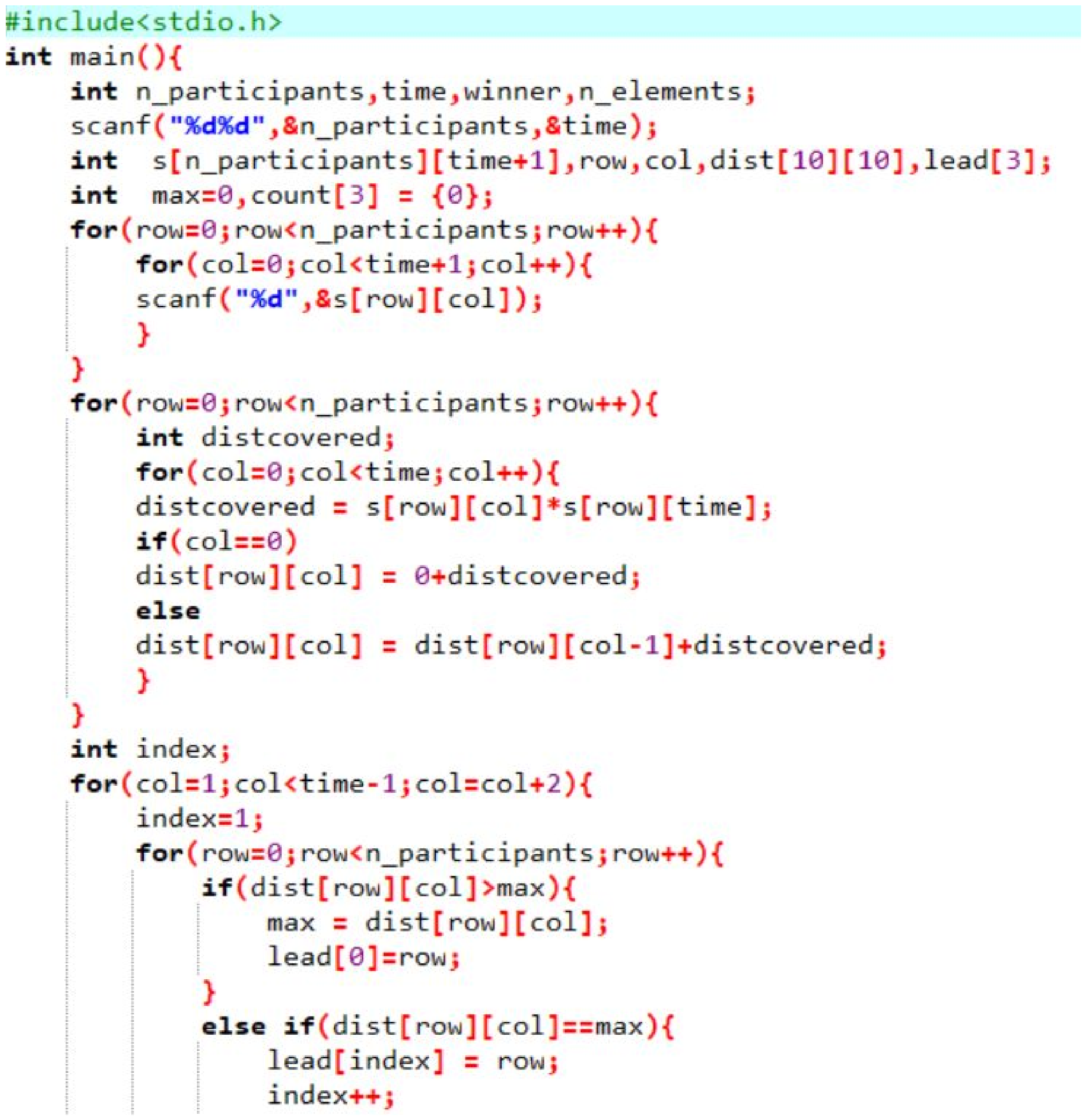
Output

```
2
```

Concepts Used to solve:-

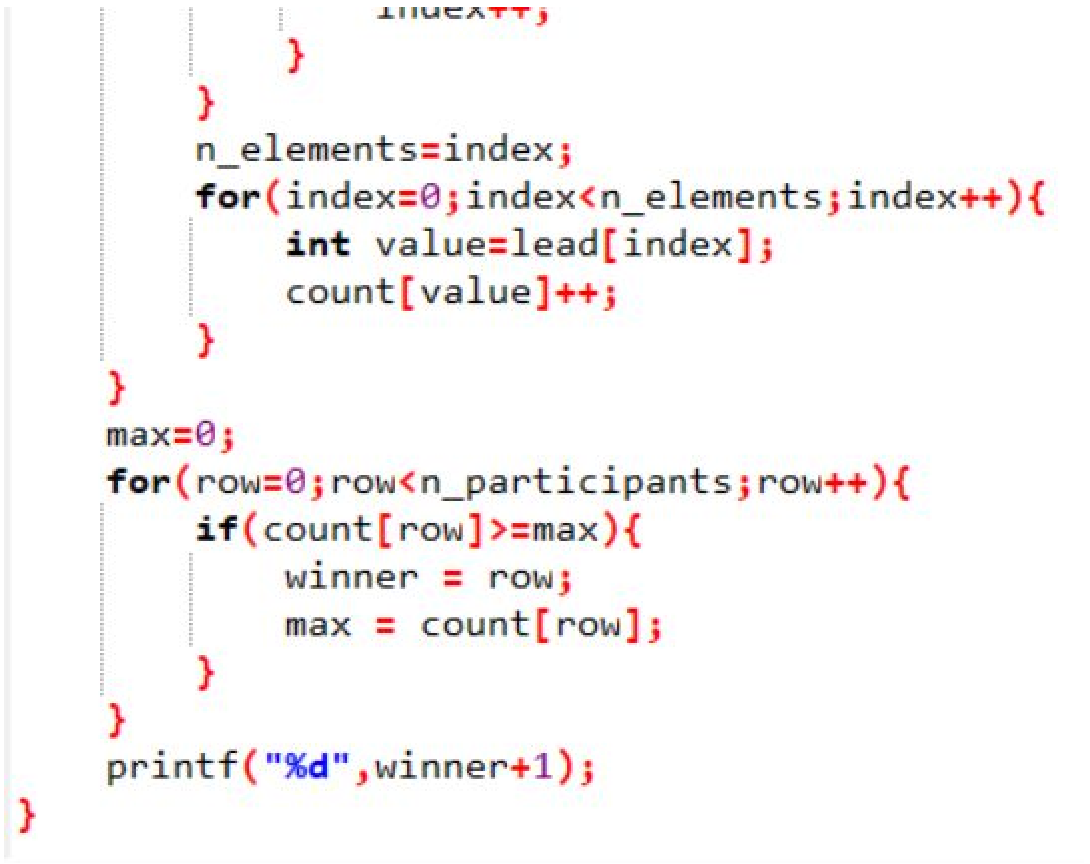
2D Arrays are used to store the number of participants for the marathon and the distance covered by them in each time interval. For loops with if conditions are used to calculate the distance covered by the runner after every time interval and calculating which participant is in the lead. Finally the participant who is in lead for more number of times will be the winner.

4.2 Coding



```
#include<stdio.h>
int main(){
    int n_participants,time,winner,n_elements;
    scanf("%d%d",&n_participants,&time);
    int  s[n_participants][time+1],row,col,dist[10][10],lead[3];
    int  max=0,count[3] = {0};
    for(row=0;row<n_participants;row++){
        for(col=0;col<time+1;col++){
        scanf("%d",&s[row][col]);
        }
    }
    for(row=0;row<n_participants;row++){
        int distcovered;
        for(col=0;col<time;col++){
        distcovered = s[row][col]*s[row][time];
        if(col==0)
        dist[row][col] = 0+distcovered;
        else
        dist[row][col] = dist[row][col-1]+distcovered;
        }
    }
    int index;
    for(col=1;col<time-1;col=col+2){
        index=1;
        for(row=0;row<n_participants;row++){
            if(dist[row][col]>max){
                max = dist[row][col];
                lead[0]=row;
            }
            else if(dist[row][col]==max){
                lead[index] = row;
                index++;
```

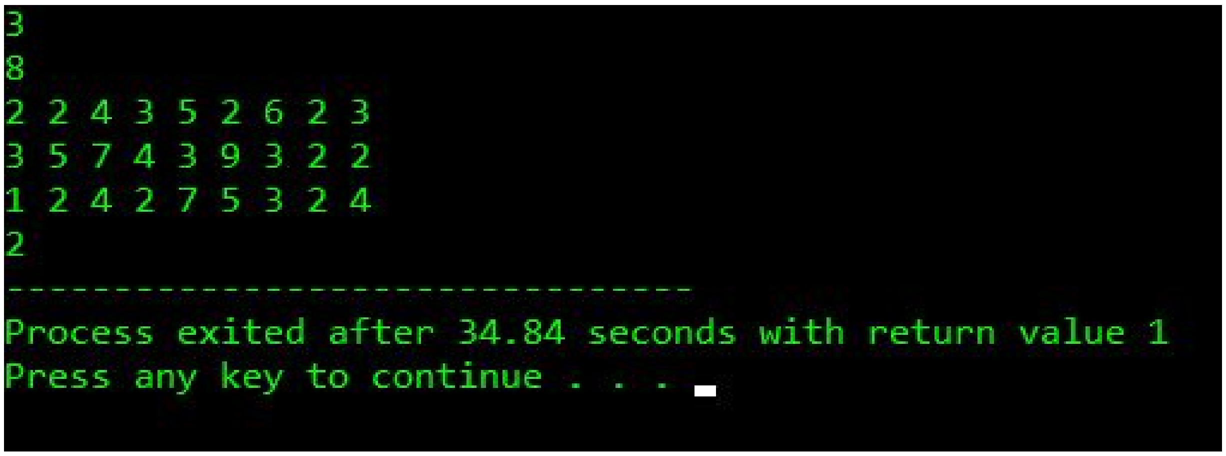
fig 4.2.1



```
                index++;
            }
        }
        n_elements=index;
        for(index=0;index<n_elements;index++){
            int value=lead[index];
            count[value]++;
        }
    }
    max=0;
    for(row=0;row<n_participants;row++){
        if(count[row]>=max){
            winner = row;
            max = count[row];
        }
    }
    printf("%d",winner+1);
}
```

fig 4.2.2

4.3 Output



```
3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4
2
--------------------------------
Process exited after 34.84 seconds with return value 1
Press any key to continue . . .
```

fig 4.3.1

5 Problem 4
Minimum Product Array

In this problem we find the minimum sum of products of 2 arrays by changing the values of 1 particular array by a given number of times.

5.1 Problem Statement:-

The task is to find the minimum sum of Products of two arrays of the same size, given that k modifications are allowed on the first array. In each modification, one array element of the first array can either be increased or decreased by 2.

Note- the product sum is Summation (A[i]*B[i]) for all i from 1 to n where n is the size of both arrays

Input Format:

3. First line of the input contains n and k delimited by whitespace
4. Second line contains the Array A (modifiable array) with its values delimited by spaces
5. Third line contains the Array B (non-modifiable array) with its values delimited by spaces

Output Format:

Output the minimum sum of products of the two arrays

Constraints:

6. $1 \leq N \leq 10^5$
7. $0 \leq |A[i]|, |B[i]| \leq 10^5$
8. $0 \leq K \leq 10^9$

Sample Input and Output

SNo.	Input	Output

1	3 5
	1 2 -3
	-2 3 -5
		-31

2	5 3
	2 3 4 5 4
	3 4 2 3 2
		25

Explanation for sample 1:

Here total numbers are 3 and total modifications allowed are 5. So we modified A[2], which is

-3 and increased it by 10 (as 5 modifications are allowed). Now final sum will be

(1 * -2) + (2 * 3) + (7 * -5)

-2 + 6 - 35

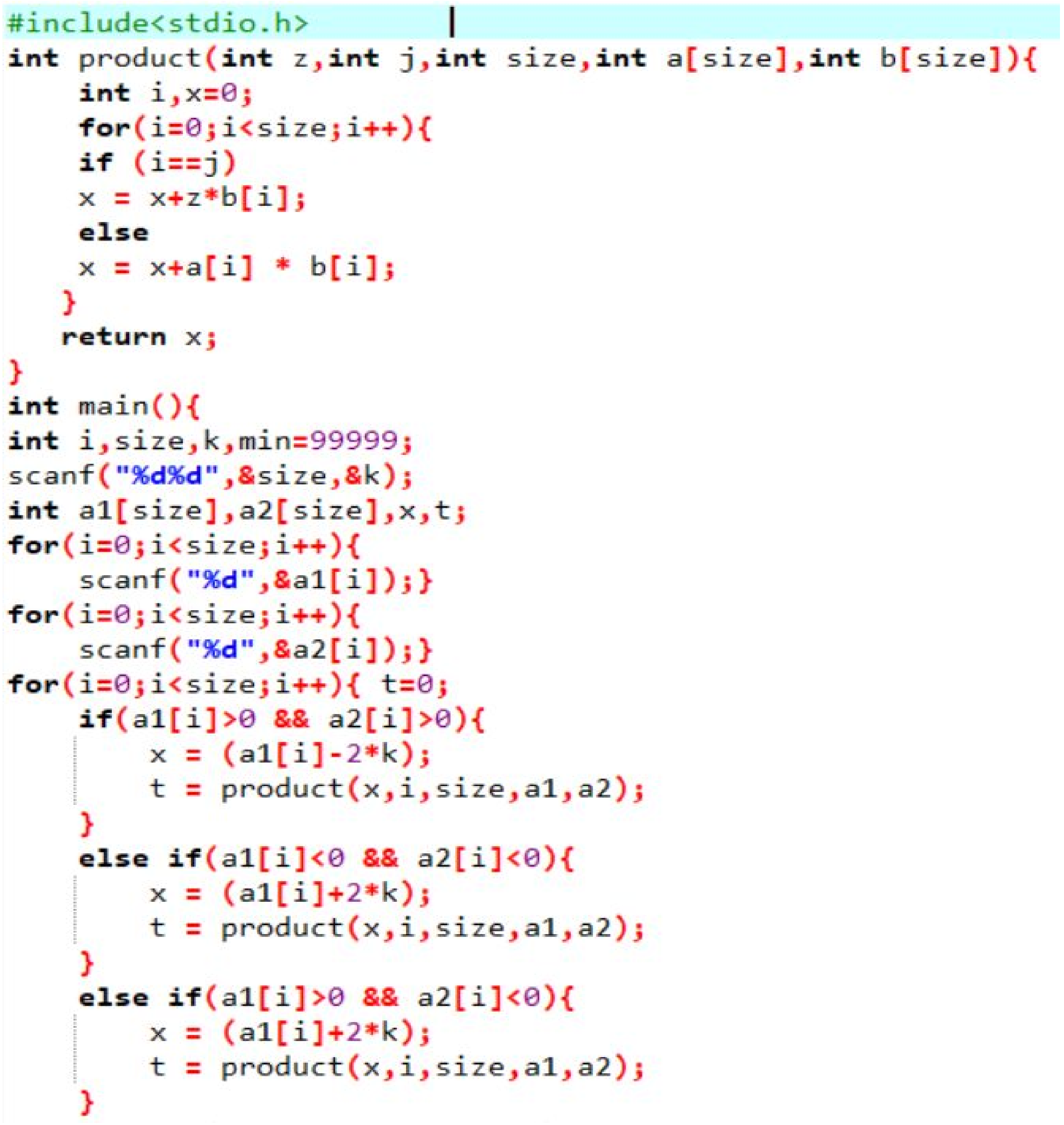
-31

-31 is our final answer.

Concepts Used To solve:-

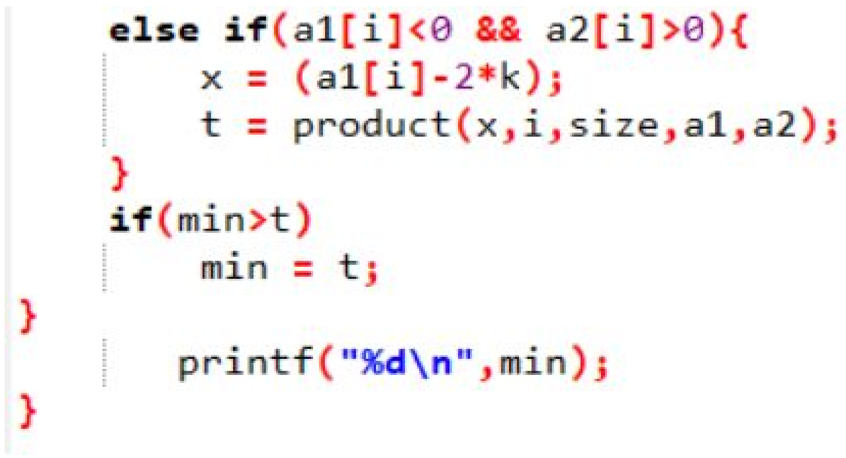
For loops with if conditions are used to find the minimum sum of products of the values of two arrays by changing one or more values of the first array by n number of times.

5.2 Coding



```
#include<stdio.h>
int product(int z,int j,int size,int a[size],int b[size]){
    int i,x=0;
    for(i=0;i<size;i++){
    if (i==j)
    x = x+z*b[i];
    else
    x = x+a[i] * b[i];
   }
   return x;
}
int main(){
int i,size,k,min=99999;
scanf("%d%d",&size,&k);
int a1[size],a2[size],x,t;
for(i=0;i<size;i++){
    scanf("%d",&a1[i]);}
for(i=0;i<size;i++){
    scanf("%d",&a2[i]);}
for(i=0;i<size;i++){ t=0;
    if(a1[i]>0 && a2[i]>0){
        x = (a1[i]-2*k);
        t = product(x,i,size,a1,a2);
    }
    else if(a1[i]<0 && a2[i]<0){
        x = (a1[i]+2*k);
        t = product(x,i,size,a1,a2);
    }
    else if(a1[i]>0 && a2[i]<0){
        x = (a1[i]+2*k);
        t = product(x,i,size,a1,a2);
    }
```

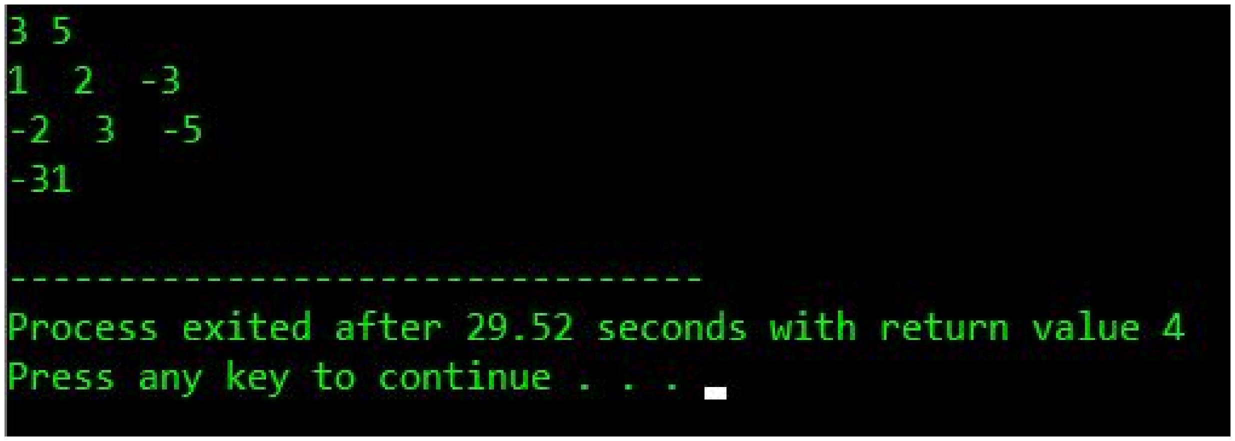
fig 5.2.1



```
    else if(a1[i]<0 && a2[i]>0){
        x = (a1[i]-2*k);
        t = product(x,i,size,a1,a2);
    }
    if(min>t)
        min = t;
}
       printf("%d\n",min);
}
```

fig 5.2.2

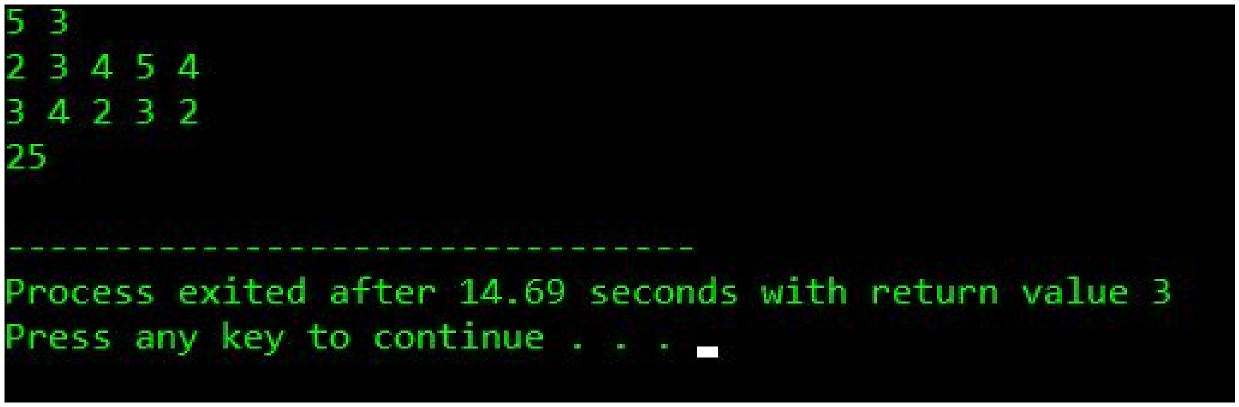
5.3 Output



```
3 5
1  2  -3
-2  3  -5
-31


--------------------------------
Process exited after 29.52 seconds with return value 4
Press any key to continue . . .
```

fig 5.3.1



```
5 3
2 3 4 5 4
3 4 2 3 2
25


--------------------------------
Process exited after 14.69 seconds with return value 3
Press any key to continue . . .
```

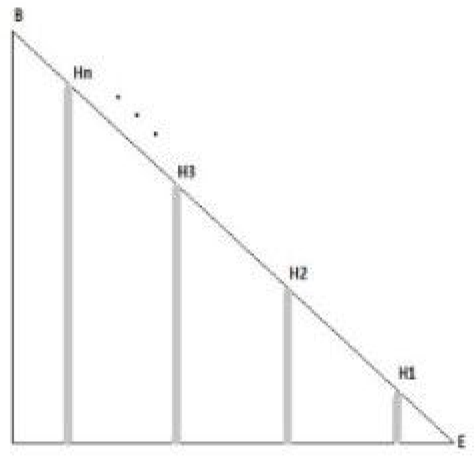
fig 5.3.2

6 Problem 5
Holes And Balls

In this problem we find the balls which will fit in the holes according to their size.

6.1 Problem Statement:-

A man is doing an experiment with the device that he built newly. The structure of the device is as below diagram.



B to E is a sloping surface with n holes, labelled H1, H2, … Hn, on it. Holes are of different diameters and depths. The man is releasing m number of balls of different diameters from the point B one after the other. He needs to find the positions of each ball after the experiment.
The specialities of the device are :

9. A ball will fall into the hole, if its diameter is less than or equal to the diameter of the hole.
10. A hole Hi will become full, if i numbers of balls fall into it. For example hole labelled H3 will become full if 3 balls fall into it.
11. If a hole is full, then no more balls fall into it.
12. A ball will reach the bottom point E from B, if and only if it is not falling into any of the holes.

Please help him in finding the eventual position of the balls. If a ball is in hole Pi, then take its position as i. If a ball reached the bottom point E, then take its position as 0.

Constraints

0 <= N <= 50
0 < Diameter of holes <= 10^9
0 < m <= 1000

0 < Diameter of balls <= 10^9
Input
Line 1 : total number of holes, N
Line 2 : N space separated integers denoting the diameters of N holes, from bottom to top.
Line 3 : total number of balls, M.
Line 4 : M space separated integers denoting diameters of balls in the order of release.
Output
Line 1 : Positions of each ball in the order of ball release separated by space.

Testcase

Input

```
3
21 3 6
11
20 15 5 7 10 4 2 1 3 6 8
```

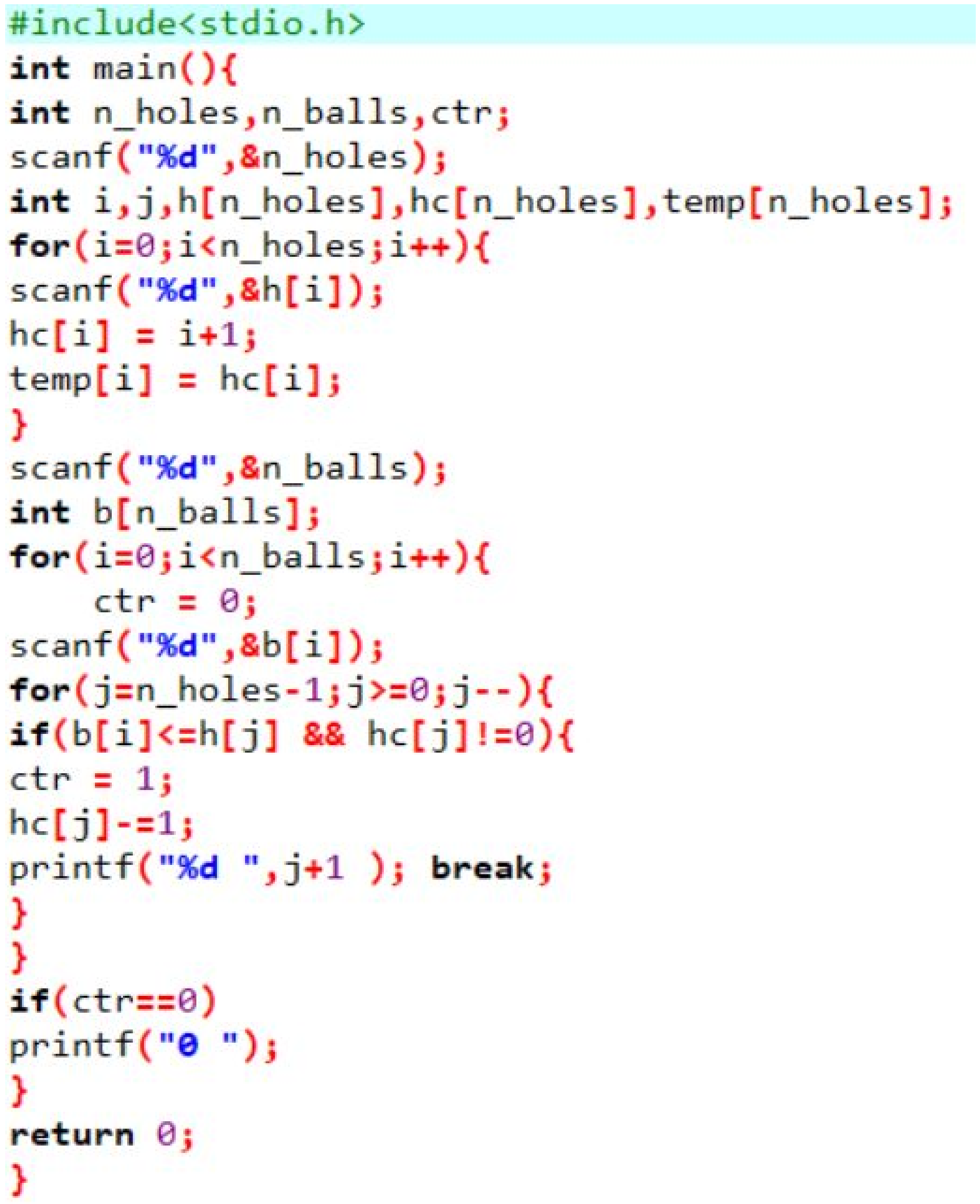
Output

```
1 0 3 0 0 3 3 2 2 0 0
```

Concepts Used to solve:-

For loops with if conditions are used in functions to check if the balls will fit in the holes or not.

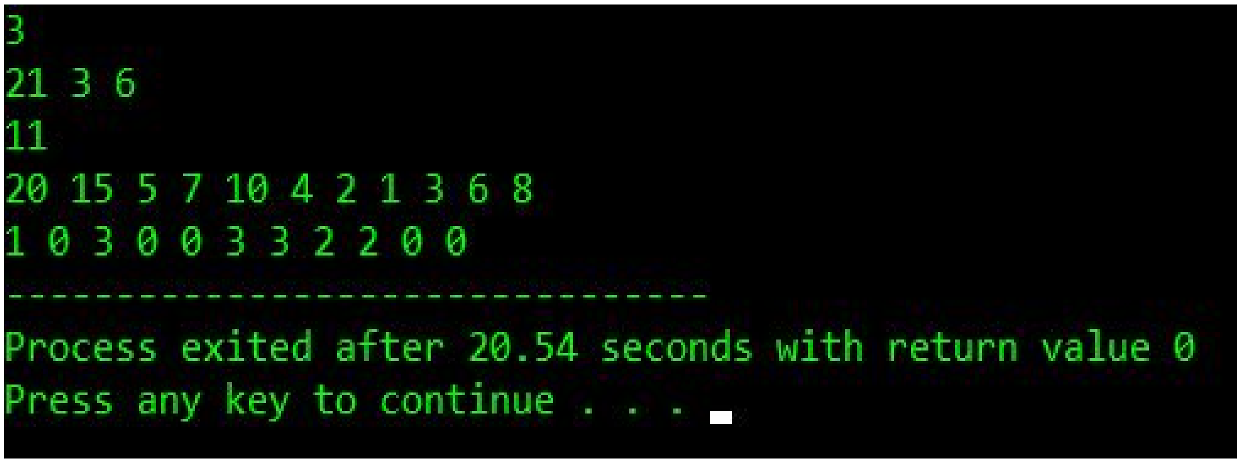
6.2Coding



```
#include<stdio.h>
int main(){
int n_holes,n_balls,ctr;
scanf("%d",&n_holes);
int i,j,h[n_holes],hc[n_holes],temp[n_holes];
for(i=0;i<n_holes;i++){
scanf("%d",&h[i]);
hc[i] = i+1;
temp[i] = hc[i];
}
scanf("%d",&n_balls);
int b[n_balls];
for(i=0;i<n_balls;i++){
    ctr = 0;
scanf("%d",&b[i]);
for(j=n_holes-1;j>=0;j--){
if(b[i]<=h[j] && hc[j]!=0){
ctr = 1;
hc[j]-=1;
printf("%d ",j+1 ); break;
}
}
if(ctr==0)
printf("0 ");
}
return 0;
}
```

fig 6.2.1

6.3 Output



```
3
21 3 6
11
20 15 5 7 10 4 2 1 3 6 8
1 0 3 0 0 3 3 2 2 0 0
--------------------------------
Process exited after 20.54 seconds with return value 0
Press any key to continue . . .
```

fig 6.3.1

7 Problem 6
Balancing Stars

In this problem we find the number of balanced brackets and if there are 2 or more stars in between every balanced brackets.

7.1 Problem Statement:-

CODU loves to play with a string of brackets. He considers string as a good string if it is balanced with stars. A string is considered as balanced with stars if the string contains balanced brackets and between every pair of bracket i.e. between opening and closing brackets, there are at least 2 stars(*) present. CODU knows how to check whether a string is balanced or not but this time he needs to keep a track of stars too. He decided to write a program to check whether a string is good or not. But CODU is not as good in programming as you are, so he decided to take help from you. Will you help him with this task? You need to print Yes and number of balanced pair if string satisfies following conditions(string is good if it satisfies following 2 conditions):

1. The string is balanced with respect to all brackets.
2. Between every pair of brackets, there are at least two stars.

However, if the string doesn't satisfy the above conditions then print No and a number of balanced pairs in a string as an output.

Constraints

4 <= String length <= 1000

Input Format

The first and only line of input contains a string of characters(a-z,A-Z), numbers(0-9), brackets('{', '[', '(', ')', ']', '}') and stars(*).

Output

Print space-separated "Yes" (without quotes) and a number of balanced pairs if the string is good. Else print "No" (without quotes) and the number of balanced pairs.

Test Case

Explanation

Example 1

Input

{**}

Output

Yes 1

Explanation

Here string contains one balanced pair {} and between this pair of the bracket, there are 2 stars present so the output is Yes with the count of balanced pair as 1.

Example 2

Input

{**(**{**[**]})}

Output

Yes 4

Explanation

A string has balanced brackets and also satisfies 2nd condition. So the output is Yes with the count of balanced pair which is 4.

Example 3

Input

**}xasd[**]sda231

Output

No 1

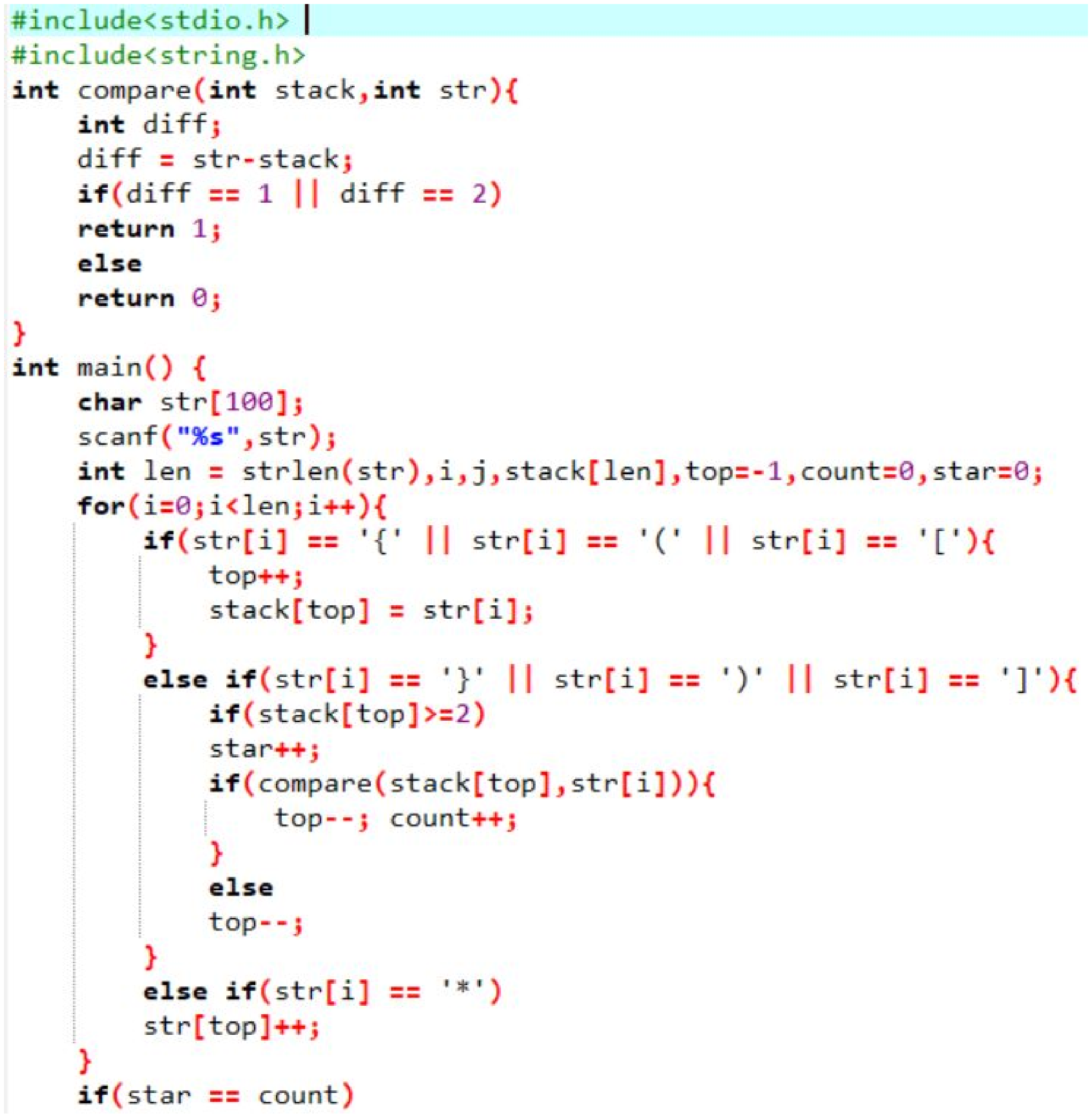
Explanation

In this case, the string is not balanced. So the output is No with the count of balanced pair as 1.

Concepts Used to solve:-

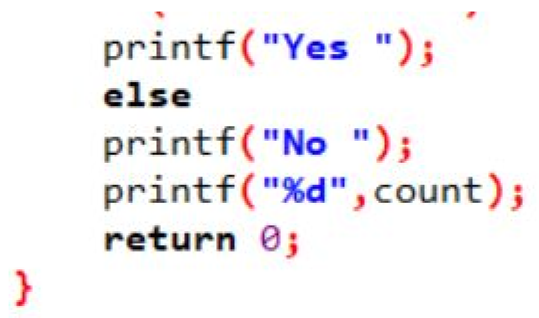
Stack is used to store open brackets in a function where for loop and if conditions are used to check the brackets and are balanced ,if so then checking if there are 2 or more number of stars in between the balanced set of brackets.

7.2 Coding



```
#include<stdio.h>
#include<string.h>
int compare(int stack,int str){
    int diff;
    diff = str-stack;
    if(diff == 1 || diff == 2)
    return 1;
    else
    return 0;
}
int main() {
    char str[100];
    scanf("%s",str);
    int len = strlen(str),i,j,stack[len],top=-1,count=0,star=0;
    for(i=0;i<len;i++){
        if(str[i] == '{' || str[i] == '(' || str[i] == '['){
            top++;
            stack[top] = str[i];
        }
        else if(str[i] == '}' || str[i] == ')' || str[i] == ']'){
            if(stack[top]>=2)
            star++;
            if(compare(stack[top],str[i])){
                top--; count++;
            }
            else
            top--;
        }
        else if(str[i] == '*')
        str[top]++;
    }
    if(star == count)
```

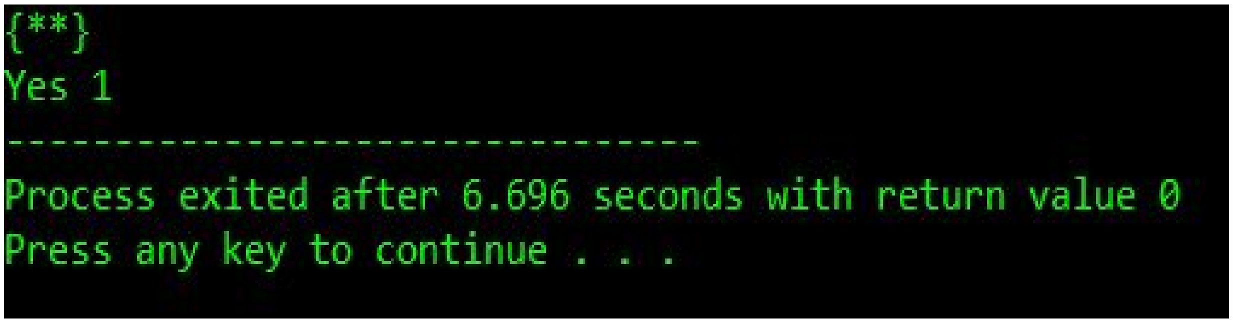
fig 7.2.1



```
    printf("Yes ");
    else
    printf("No ");
    printf("%d",count);
    return 0;
}
```

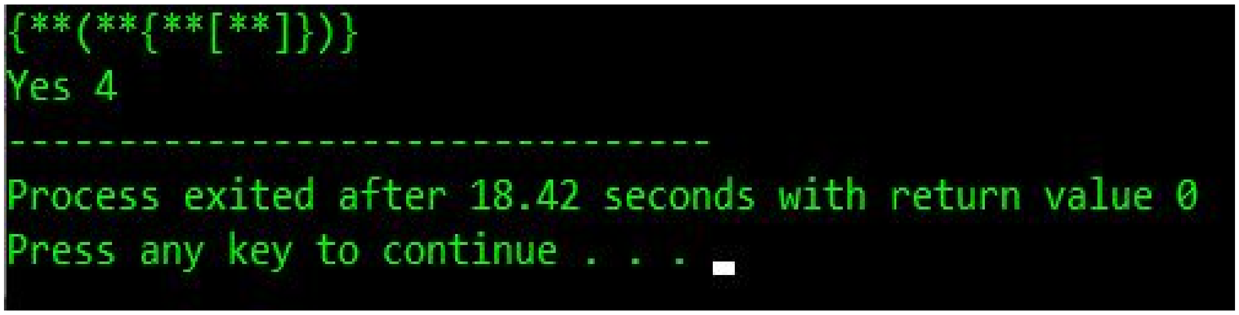
fig 7.2.2

7.3 Output



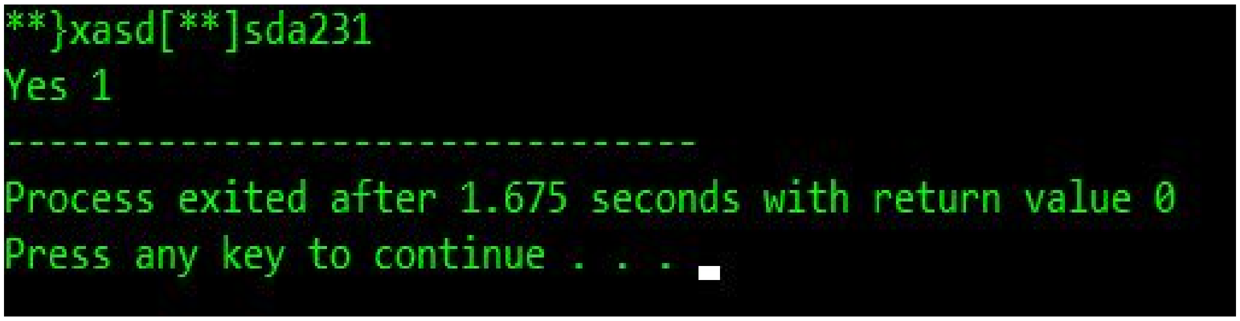
```
{**}
Yes 1
--------------------------------
Process exited after 6.696 seconds with return value 0
Press any key to continue . . .
```

fig 7.3.1



```
{**(**{**[**]})}
Yes 4
--------------------------------
Process exited after 18.42 seconds with return value 0
Press any key to continue . . .
```

fig 7.3.2



```
**}xasd[**]sda231
Yes 1
--------------------------------
Process exited after 1.675 seconds with return value 0
Press any key to continue . . .
```

fig 7.3.3

8 Problem 7
Grooving Monkeys

In this problem we find the time taken by monkeys to come to their initial position where they are following a particular pattern while dancing in a party.

8.1 Problem Statement:-

N monkeys are invited to a party where they start dancing. They dance in a circular formation, very similar to a Gujarati Garba or a Drum Circle. The dance requires the monkeys to constantly change positions after every 1 second.
The change of position is not random & you, in the audience, observe a pattern. Monkeys are very disciplined & follow a specific pattern while dancing.
Consider N = 6, and an array monkeys = {3,6,5,4,1,2}.
This array (1-indexed) is the dancing pattern. The value at monkeys[i], indicates the new of position of the monkey who is standing at the ith position.
Given N & the array monkeys[ ], find the time after which all monkeys are in the initial positions for the 1st time.

Constraints

1<=t<=10 (test cases)
1<=N<=10000 (Number of monkeys)

Input Format

First line contains single integer t, denoting the number of test cases.
Each test case is as follows -
Integer N denoting the number of monkeys.
Next line contains N integer denoting the dancing pattern array, monkeys[].

Output

t lines,
Each line must contain a single integer T, where T is the minimum number of seconds after which all the monkeys are in their initial position.

Test Case

Explanation

Example 1
Input
1

6

3 6 5 4 1 2

Output
6

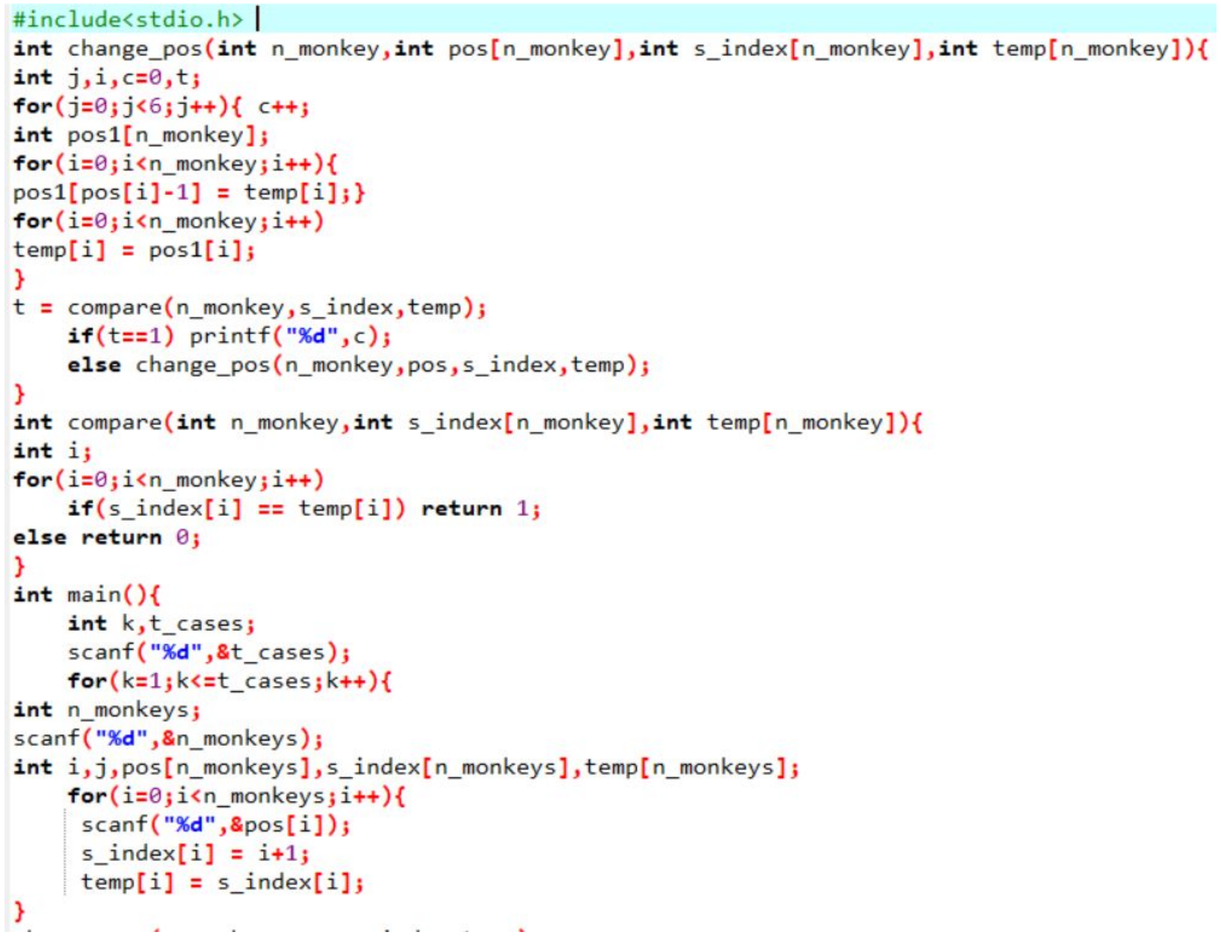
Explanation

Consider N = 6, and an array monkeys = {3,6,5,4,1,2}.
Suppose monkeys are a,b,c,d,e,f, & Initial position (at t = 0) -> a,b,c,d,e,f
At t = 1 -> e,f,a,d,c,b
a will move to 3rd position, b will move to 6th position, c will move to 5th position, d will move to 4th position, e will move to 1st position and f will move to 2nd position. Thus from a,b,c,d,e,f at t =0, we get e,f,a,d,c,b at t =1. Recursively applying same transpositions, we get following positions for different values of t.
At t = 2 -> c,b,e,d,a,f
At t = 3 -> a,f,c,d,e,b
At t = 4 -> e,b,a,d,c,f
At t = 5 -> c,f,e,d,a,b
At t = 6 -> a,b,c,d,e,f
Since at t = 6, we got the original position, therefore the answer is 6.

Concepts Used to solve:-

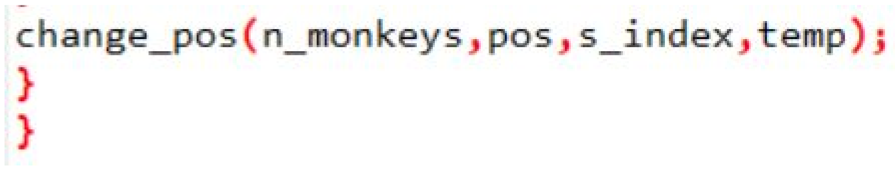
Arrays are used to store the places of monkeys and for loops and if conditions are used in function to count the number of seconds needed by the monkeys to come to their original positions.

8.2Coding



```c
#include<stdio.h>
int change_pos(int n_monkey,int pos[n_monkey],int s_index[n_monkey],int temp[n_monkey]){
int j,i,c=0,t;
for(j=0;j<6;j++){ c++;
int pos1[n_monkey];
for(i=0;i<n_monkey;i++){
pos1[pos[i]-1] = temp[i];}
for(i=0;i<n_monkey;i++)
temp[i] = pos1[i];
}
t = compare(n_monkey,s_index,temp);
    if(t==1) printf("%d",c);
    else change_pos(n_monkey,pos,s_index,temp);
}
int compare(int n_monkey,int s_index[n_monkey],int temp[n_monkey]){
int i;
for(i=0;i<n_monkey;i++)
    if(s_index[i] == temp[i]) return 1;
else return 0;
}
int main(){
    int k,t_cases;
    scanf("%d",&t_cases);
    for(k=1;k<=t_cases;k++){
int n_monkeys;
scanf("%d",&n_monkeys);
int i,j,pos[n_monkeys],s_index[n_monkeys],temp[n_monkeys];
    for(i=0;i<n_monkeys;i++){
     scanf("%d",&pos[i]);
     s_index[i] = i+1;
     temp[i] = s_index[i];
}
```

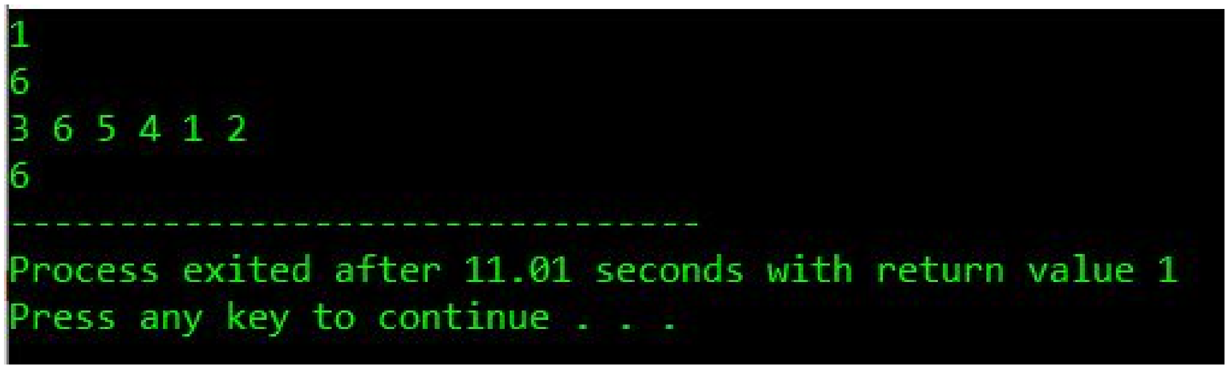
fig 8.2.1



```c
change_pos(n_monkeys,pos,s_index,temp);
}
}
```

fig 8.2.2

8.3 Output



```
1
6
3 6 5 4 1 2
6
--------------------------------
Process exited after 11.01 seconds with return value 1
Press any key to continue . . .
```

fig 8.3.1

9 Problem 8
Consecutive Prime Sum

In this problem we find the count of prime numbers which can be expressed as a sum of other consecutive prime numbers.

9.1 Problem Statement:-

Some prime numbers can be expressed as a sum of other consecutive prime numbers.

- For example

    - 5 = 2 + 3,
    - 17 = 2 + 3 + 5 + 7,
    - 41 = 2 + 3 + 5 + 7 + 11 + 13.

      Your task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.

Write code to find out the number of prime numbers that satisfy the above-mentioned property in a given range.

Input Format: First line contains a number N

Output Format: Print the total number of all such prime numbers which are less than or equal to N.

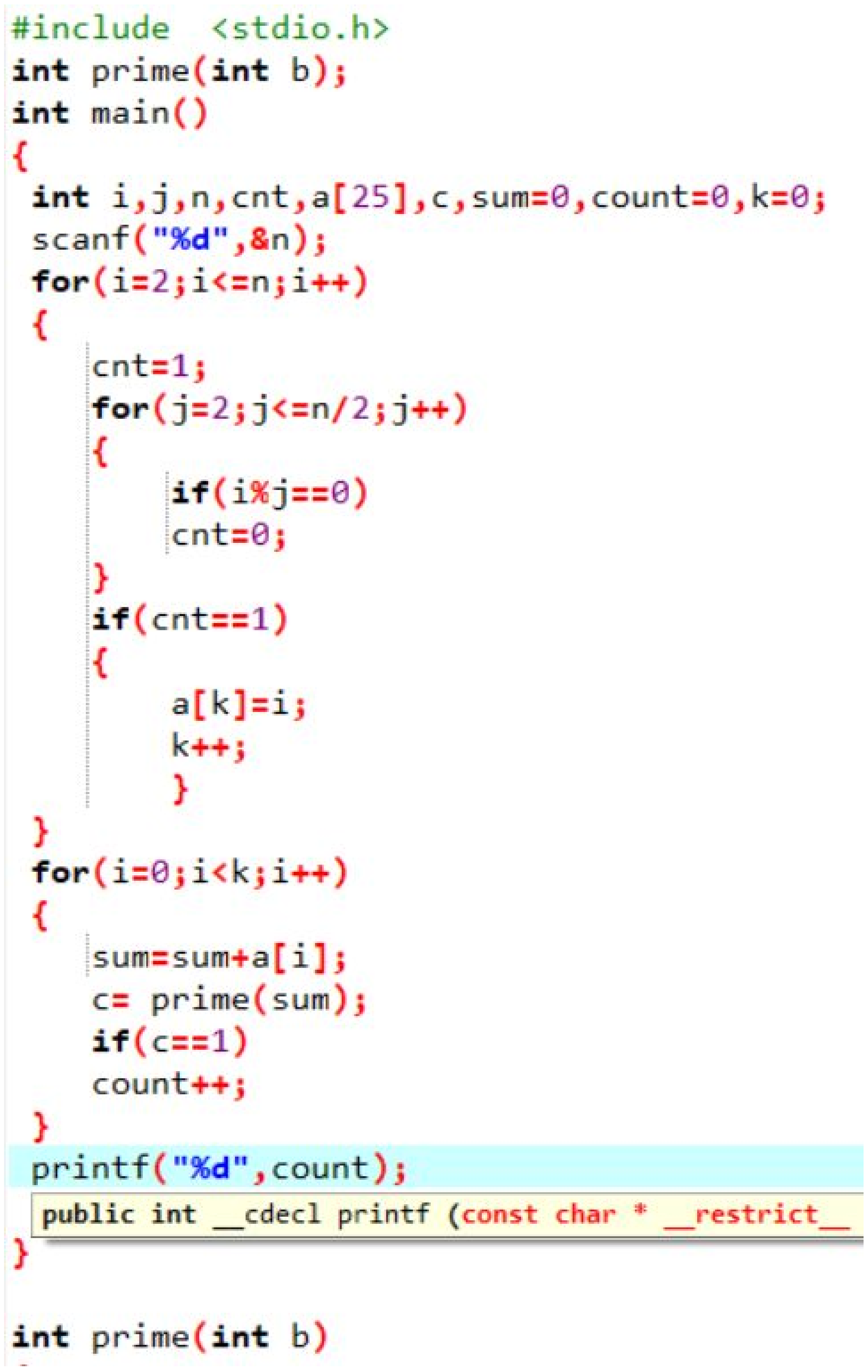
Constraints: 2<N<=12,000,000,000

S.no	Input	Output	Comment
1	**20**	**2**	**Below 20 there are two such members; 5 and 17. 5=2+3 17=2+3+65+7**
2	5		

Concepts Used to solve:

For loops and if conditions to check if the number is prime and if so then add them to get a prime

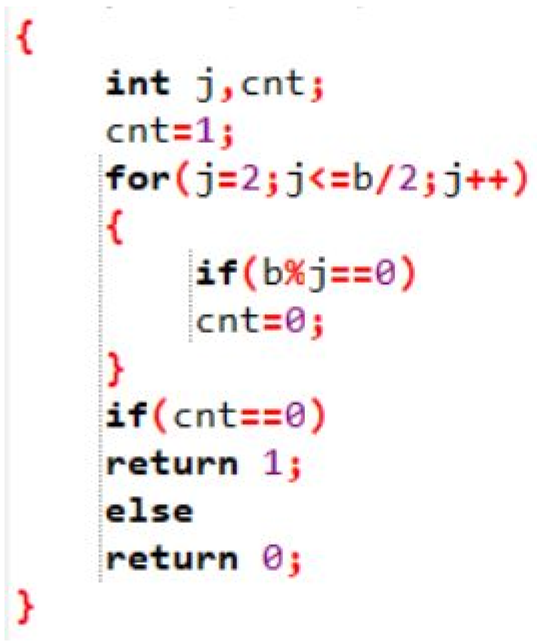
9.2Coding



```
#include  <stdio.h>
int prime(int b);
int main()
{
 int i,j,n,cnt,a[25],c,sum=0,count=0,k=0;
 scanf("%d",&n);
 for(i=2;i<=n;i++)
 {
     cnt=1;
     for(j=2;j<=n/2;j++)
     {
         if(i%j==0)
         cnt=0;
     }
     if(cnt==1)
     {
         a[k]=i;
         k++;
         }
 }
 for(i=0;i<k;i++)
 {
     sum=sum+a[i];
     c= prime(sum);
     if(c==1)
     count++;
 }
 printf("%d",count);
 public int __cdecl printf (const char * __restrict__
}

int prime(int b)
```

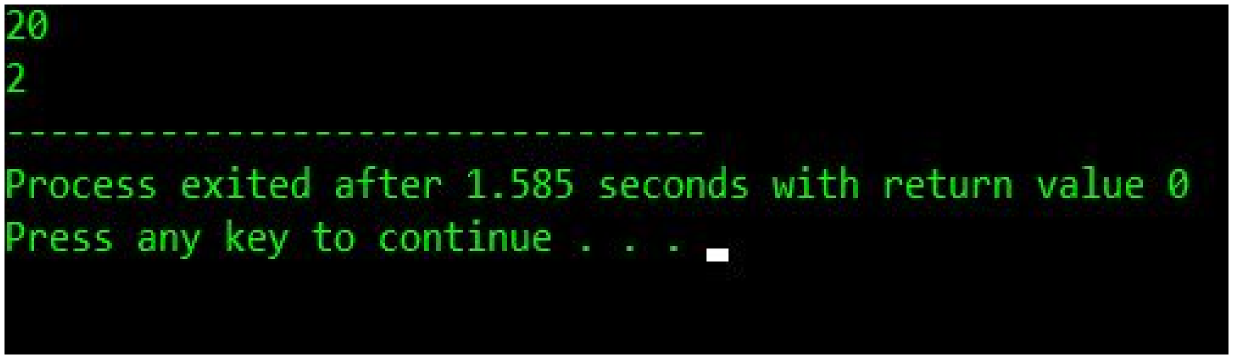
fig 9.2.1



```
{
	int j,cnt;
	cnt=1;
	for(j=2;j<=b/2;j++)
	{
		if(b%j==0)
		cnt=0;
	}
	if(cnt==0)
	return 1;
	else
	return 0;
}
```

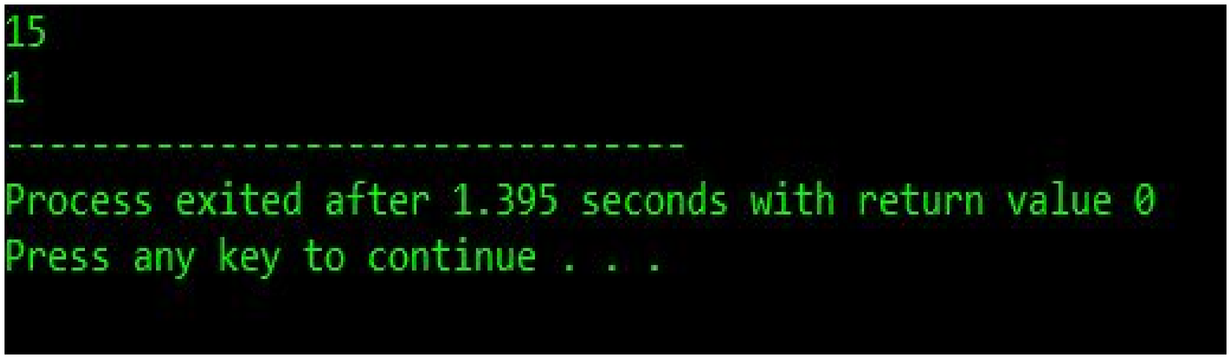
fig 9.2.2

9.3 Output



```
20
2
--------------------------------
Process exited after 1.585 seconds with return value 0
Press any key to continue . . .
```

fig 9.3.1



```
15
1
--------------------------------
Process exited after 1.395 seconds with return value 0
Press any key to continue . . .
```

fig 9.3.2

10 Software Requirements

10.1 Hardware Requirements

This project can be executed in any system or an android phone without prior to any platform. We can use any online compiler and interpreter.

10.2 Software Requirements

There are two ways to execute this projects

- Online compilers
- Softwares for execution (DEV C++,)

Online Compilers require only internet connection. We have many free compilers with which we can code.Softwares for execution need to be installed based on the user's system specification. These help us to completely execute the project. These softwares are based on the platforms.

11 BIBLIOGRAPHY

- https://www.programminggeek.in/2013/08/solution-of-game-of-primes-problem-of-round1-of-codevita2013-organized-by-TCS.html#.Xx1ytp4zZPY
- https://codeofgeeks.com/television-sets/
- https://codeofgeeks.com/marathon-winner/
- http://codepiggy.blogspot.com/2018/08/minimum-product-array.html
- https://codeofgeeks.com/holes-and-balls/
- https://codeofgeeks.com/balancing-stars/
- https://codeofgeeks.com/grooving-monkeys/
- https://prepinsta.com/tcs-codevita/c-code-for-consecutive-prime-sum/