**18 Coding questions**

Q.1. Prime Numbers

A school students want to check prime numbers within given range, write a c program to help them for the same. Where x will be starting range and y will be ending range, if x is greater than y or x is equal to y print 0, if x is smaller than y print the prime numbers within mentioned range.

A number is said to be prime if it is divisible by 1 and the number itself.

**Sample Input 1**

50 90

**Sample Output 1**

53 59 61 67 71 73 79 83 89

**Sample Input 2**

60 6

**Sample Output 2**

0

**Input Explanation**

Input consists of two space separated integer value

First input will be taken as the value for x that is starting value

Second input will be taken as the value for y that is ending value

**Output Explanation**

Output can consists multiple space seperated integer value based upon input

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 100 150 | 45 7 | 15 60 | 55 3 | 25 25 |
| **Output** | 101 103 107 109 113 127 131 137 139 149 | 0 | 17 19 23 29 31 37 41 43 47 53 59 | 0 | 0 |

**#Solution**

#include <stdio.h>

int main()

{

int x, y, i, flag;

scanf("%d %d",&x, &y);

if (x > y || x==y)

{

printf("0");

}

else

{

while (x < y)

{

flag = 0;

for (i = 2; i <= x / 2; ++i)

{

if (x % i == 0)

{

flag = 1;

break;

}

}

if (flag == 0)

printf("%d ", x);

++x;

}

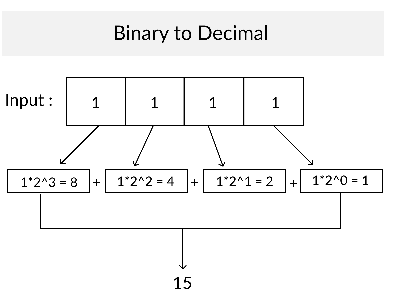
}

return 0;

}

Q.2. Binary to Decimal conversion

Ajit is working in IT company and he is getting binary values from computers, as a human he is not able understand binary values, so help Ajit to understand binary values by converting it into decimal values.



Consider the binary number from the last.

For the above mentioned example,

1 \* 2^0 = 1

1 \* 2^1 = 2

1 \* 2^2 = 4

1 \* 2^3 = 6

Decimal number = 1 + 2 + 4 + 6 = 15,

1111 in binary form is represented as 15 in decimal.

**Sample Input 1**

101011

**Sample Output 1**

43

**Sample Input 2**

110110

**Sample Output 2**

54

**Input Explanation**

Input consists of single integer value

**Output Explanation**

Output consists of single integer value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 111111111 | 100011 | 1110011 | 1011001 | 100100100 |
| **Output** | 511 | 35 | 115 | 89 | 292 |

**#Solution**

#include <stdio.h>

#include<math.h>

int binary\_to\_decimal(long int n)

{

int decimal = 0, i = 0, remainder;

while (n != 0)

{

remainder = n % 10;

n /= 10;

decimal += remainder \* pow(2, i);

++i;

}

return decimal;

}

int main()

{

long int n;

scanf("%ld", & n);

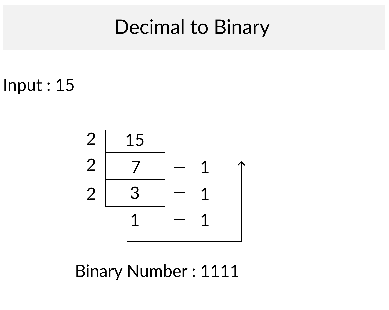
printf("%d",binary\_to\_decimal(n));

return 0;

}

Q.3. Decimal to Binary conversion

Computer cannot understand decimal numbers, it understands only zeros and ones. Write a program to help computer by converting decimal number to binary.



Considering the same example,

15 / 2 = 7 rem = 1,

7 / 2 = 3 rem = 1,

3 / 2 = 1 rem = 1,

1 / 2 = 0 rem = 1

Binary equivalent of 15 is 1111.

**Sample Input 1**

100

**Sample Output 1**

1100100

**Sample Input 2**

215

**Sample Output 2**

11010111

**Input Explanation**

Input consists of single integer value

**Output Explanation**

Output consists of single integer value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 334 | 25 | 99 | 123 | 486 |
| **Output** | 101001110 | 11001 | 1100011 | 1111011 | 111100110 |

**#Solution**

#include <stdio.h>

long int decimal\_to\_binary(int n)

{

long int binary = 0;

int remainder, i, flag = 1;

for (i = 1; n != 0; i = i \* 10)

{

remainder = n % 2;

n /= 2;

binary += remainder \* i;

}

return binary;

}

int main()

{

int n;

scanf("%d", &n);

printf("%d",decimal\_to\_binary(n));

return 0;

}

Q.4. Number of Handshakes

It was Raj's first day at school. His teacher Anu asked the students to meet every other student in the class and to introduce about themselves. The teacher asked them to do handshakes when they meet each other.

If there are n number of students in the class then find the total number of handshakes made by the students.

• Input the number of people (n).

• Find nC2, calculated as n \* (n-1) / 2.

• Print the calculated result.

**Sample Input 1**

15

**Sample Output 1**

105

**Sample Input 2**

10

**Sample Output 2**

45

**Input Explanation**

Input consists of one integer , which corresponds to the total number of students.

**Output Explanation**

Output consists of one integer, which corresponds to the total number of handshakes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 25 | 11 | 17 | 55 | 22 |
| **Output** | 300 | 55 | 136 | 1485 | 231 |

**#Solution**

#include<stdio.h>

int main()

{

int num;

scanf("%d", &num);

int total = num \* (num - 1) / 2;

printf("%d", total);

return 0;

}

Q.5. Program to find all possible permutations in which 'n' people can occupy 'r' seats in a theatre.

N friends are planning to go to a movie. One among them suggested few movies and all others started to discuss and finally they selected a movie. One among them quickly booked their tickets online, to their surprise they are unable to select their seats. All of them got confused. Then anyhow, decided to go to the movie. They rushed to reach the theatre on time. Again, they are surprised that no one was there in the theatre. They are the only people about to watch the movie. There is 'r' number of seats in which, 'n' number persons should sit. In how many ways they can sit inside the theatre?

Given the number of people 'n' and the number of seats 'r' as input. The task is to find the different number of ways in which 'n' number of people can be seated in those 'r' number of seats.

For example,

Input:

Number of people: 5

Number of Rows: 3

Output:

The total number of ways in which 'n' people can be seated in 'r' seats = 60.

Calculation:

P(n,r) =P(5,3)

=5! /(5?3)! = 5! / ( 5 ? 3 )!

= 120 / 2 = 60

**Sample Input 1**

5

3

**Sample Output 1**

60

**Sample Input 2**

7

3

**Sample Output 2**

210

**Input Explanation**

Input consists of two integer value

First input will be number of people

Second input will be number of Rows

**Output Explanation**

Output consists of single integer value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 12  1 | 11  4 | 9  3 | 6  2 | 4  6 |
| **Output** | 12 | 7920 | 504 | 30 | 360 |

**#Solution**

#include<stdio.h>

int fact(long int x)

{

long int f = 1, i;

for (i = 1; i <= x; i++)

{

f = f \* i;

}

return f;

}

int main()

{

long int n, r, p, temp;

long int num, den;

scanf("%ld",&r);

scanf("%ld",&n);

if (n < r)

{

temp = n;

n = r;

r = temp;

}

num = fact(n);

den = fact(n - r);

p = num / den;

printf("%ld", p);

}

Q.6. Occurrence of digit

Write a program to find the number of times digit m occurs in each and every number from 0 to n. Given a number n as input, count the number of m’s occurring in range from 0 to n. (value of range will be from 0 to n)

For example,

Input1: 100

Input2: 3

Output: 20

Total number of 3s that appear from numbers 0 to 100 are {3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43, 53, 63, 73, 83, 93}

**Sample Input 1**

50

5

**Sample Output 1**

6

**Sample Input 2**

45

2

**Sample Output 2**

15

**Input Explanation**

Input consists of two integer value

First input is ending value of the range

Second input is the single digit value to find the occurrence

**Output Explanation**

Output consists of integer value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 99  2 | 75  1 | 88  3 | 15  2 | 27  7 |
| **Output** | 20 | 17 | 19 | 2 | 3 |

**#Solution**

#include <stdio.h>

int counts(int n, int m)

{

int count = 0;

while (n > 0)

{

if (n % 10 == m)

{

count++;

}

n = n / 10;

}

return count;

}

int count\_in\_range(int n,int m)

{

int count = 0;

for (int i = 2; i <= n; i++)

{

count += counts(i,m);

}

return count;

}

int main()

{

int n,m;

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

printf("%d",count\_in\_range(n,m));

return 0;

}

Q.7. Number palindrome pattern

To print palindrome pyramid pattern using numbers is discussed here. Given a number n, the task is to print a palindrome pyramid containing n number of rows.

**Sample Input 1**

5

**Sample Output 1**

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

**Sample Input 2**

3

**Sample Output 2**

1

121

12321

**Input Explanation**

Input consists of single integer value

**Output Explanation**

Output consists of number palindrome pattern depending on the input

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 3 | 4 | 6 | 2 | 5 |
| **Output** | 1  121  12321 | 1  121  12321  1234321 | 1  121  12321  1234321  123454321  12345654321 | 1  121 | 1  121  12321  1234321  123454321 |

**#Solution**

#include<stdio.h>

int main()

{

int i, j, k, l, n;

scanf("%d", &n);

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

{

for (k = 1; k <= i; k++)

{

printf("%d",k);

}

for (l = i - 1; l >= 1; l--)

{

printf("%d",l);

}

printf("\n");

}

return 0;

}

Q.8. String sorting

Given a string, the task is to sort the string in alphabetical order and display it as output.

**Sample Input 1**

face

**Sample Output 1**

acef

**Sample Input 2**

focus

**Sample Output 2**

cfosu

**Input Explanation**

Input consists of character value

**Output Explanation**

Output consists of sorted character value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | Complete | return | hello | Welcome | Breakfast |
| **Output** | Ceelmopt | enrrtu | ehllo | Wceelmo | aabefkrst |

**#Solution**

#include <stdio.h>

#include <string.h>

int main()

{

char string[100];

scanf("%s", string);

char temp;

int i, j;

int n = strlen(string);

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

{

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

{

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

{

temp = string[i];

string[i] = string[j];

string[j] = temp;

}

}

}

printf("%s", string);

return 0;

}

Q.9. Print only alphabet

A string is obtained as input from the user and all the characters other than the alphabets are removed from the string and the output string containing only the alphabets is displayed.

**Sample Input 1**

We23lc333om@#e

**Sample Output 1**

Welcome

**Sample Input 2**

h@#el#$lo

**Sample Output 2**

hello

**Input Explanation**

Input consists of string value

**Output Explanation**

Output consists of string value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | my@anatomy | Chi123\*()tkara | Wel32439(\*come | Univer90\*()sity | Hor23)\*s23e |
| **Output** | Myanatomy | Chitkara | Welcome | University | Horse |

**#Solution**

#include<stdio.h>

int main()

{

char input[150];

int i, j;

gets(input);

for (i = 0; input[i] != '\0'; ++i)

{

while (!((input[i] >= 'a' && input[i] <= 'z') || (input[i] >= 'A' && input[i] <= 'Z') || input[i] == '\0'))

{

for (j = i; input[j] != '\0'; ++j)

{

input[j] = input[j + 1];

}

input[j] = '\0';

}

}

puts(input);

return 0;

}

Q.10. Occurrence of character

Given a string, the occurrence of each character is displayed as output in sorted form.

**Sample Input 1**

google

**Sample Output 1**

e 1

g 2

l 1

o 2

**Sample Input 2**

America

**Sample Output 2**

A 1

a 1

c 1

e 1

i 1

m 1

r 1

**Input Explanation**

Input consists of string value

**Output Explanation**

Output consists of space separated character and integer value, depending on the input

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | Hello | MyAnatomy | Microsoft | sitting | kangaroo |
| **Output** | H 1  e 1  l 2  o 1 | A 1  M 1  a 1  m 1  n 1  o 1  t 1  y 2 | M 1  c 1  f 1  i 1  o 2  r 1  s 1  t 1 | g 1  i 2  n 1  s 1  t 2 | a 2  g 1  k 1  n 1  o 2  r 1 |

**#Solution**

#include <stdio.h>

#include <string.h>

int main()

{

char str[100];

int i;

int freq[256] = {0};

gets(str);

for (i = 0; str[i] != '\0'; i++)

{

freq[str[i]]++;

}

for (i = 0; i < 256; i++)

{

if (freq[i] != 0)

{

printf("%c %d\n", i, freq[i]);

}

}

return 0;

}

Q.11. Anagram or not.

Two strings are given as input and those strings have to be checked if they are anagrams or not. Anagram means that both strings contain the same character set, only their order will be different. Therefore, in both strings, the frequency of each letter must be the same.

**Sample Input 1**

act

cat

**Sample Output 1**

Anangram

**Sample Input 2**

team

mat

**Sample Output 2**

Not Anagrams

**Input Explanation**

Input consists of two line separated string value (in lowercase only)

**Output Explanation**

Output consists of single string value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | cat  bat | rat  art | den  end | dead  bread | tab  bat |
| **Output** | Not Anagrams | Anagram | Anagrams | Not Anagrams | Anagrams |

**#Solution**

#include <stdio.h>

int check\_anagram(char[], char[]);

int main()

{

char a[100], b[100];

scanf("%s", a);

scanf("%s", b);

if (check\_anagram(a, b) == 1)

printf("The strings are anagrams\n");

else

printf("The strings are not anagrams\n");

return 0;

}

int check\_anagram(char a[], char b[])

{

int first[26] = {0}, second[26] = {0}, c = 0;

while (a[c] != '\0')

{

first[a[c] - 'a']++;

c++;

}

c = 0;

while (b[c] != '\0')

{

second[b[c] - 'a']++;

c++;

}

for (c = 0; c < 26; c++)

{

if (first[c] != second[c])

return 0;

}

return 1;

}

Q.12. Count odd and even.

Given an array of integers, count the total number of odd elements and even elements in the array and display them as output.

**Sample Input 1**

3

1 2 3

**Sample Output 1**

Odd: 2

Even: 1

**Sample Input 2**

5

1 2 3 4 5

**Sample Output 2**

Odd: 3

Even: 2

**Input Explanation**

Input consists of two lines

First line input represents size of array

Second line will be multiple space separate integer value, that are elements of array

**Output Explanation**

Output consists combination of character and integers in two lines

First line will display count of even

Second line will display count of odd numbers

**Note: make sure there are no extra spaces in output line**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 4  12 15 16 1 | 6  99 65 1 3 7 33 | 3  33 22 11 | 4  33 74 89 66 | 5  11 2 4 15 8 |
| **Output** | Odd: 2  Even: 2 | Odd: 6  Even: 0 | Odd: 2  Even: 1 | Odd: 2  Even: 2 | Odd: 2  Even: 3 |

**#Solution**

#include<stdio.h>

int main()

{

int n;

scanf("%d", &n);

int arr[n];

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

{

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

}

int count\_odd = 0, count\_even = 0;

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

{

if (arr[i] % 2 == 1)

count\_odd++;

else

count\_even++;

}

printf("Odd: %d", count\_odd);

printf("\nEven: %d", count\_even);

return 0;

}

Q.13. The online math course provided 'MathAtTip' has designed a course for children called Learning Number Recognition and Counting. The assessment part of the course has a question where the student is given a number and a digit. The student needs to find out the total count of the digits present in the number excluding the given digit.

Write a c program to help the student find out the count of the total number of digits present in the number excluding the given digit.

**Sample Input 1**

5644456 5

**Sample Output 1**

5

**Sample Input 2**

55555 5

**Sample Output 2**

0

**Input Explanation**

The input consists of two space-separated integers – number and digit,

where the first integer represents the number and the second integer represents the digit given to the student.

**Output Explanation**

Print an integer representing the count of the total number of digits present in the number excluding the given digit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 775642156 6 | 23145645 7 | 443475 4 | 5644456 5 | 111111111 1 |
| **Output** | 7 | 8 | 3 | 5 | 0 |

**#Solution**

#include<stdio.h>

int excludingDigit(int,int);

int main()

{

int num,n;

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

printf("%d",excludingDigit(num,n));

return 0;

}

int excludingDigit(int num,int n)

{

int count=0,digit;

while(num)

{

digit=num%10;

num=num/10;

if(digit!=n)

{

count++;

}

}

return count;

}

Q.14. Shooting Game

The games development company "FunGames" has developed a balloon shooter game. The balloons are arranged in a linear sequence and each balloon has a number associated with it. The numbers on the balloons are in the Fibonacci series. In the game, the player shoots 'k' balloons. The player's score is the sum of numbers on the 'k' balloons. Write a program to generate the player's score.

Example

Input

7

Output

20

Explanation

The Fibonacci sum is 0+1+1+2+3+5+8=20

**Sample Input 1**

8

**Sample Output 1**

33

**Sample Input 2**

0

**Sample Output 2**

0

**Input Explanation**

The input consists of an single integer, representing the total number of balloons shot by the player (k).

**Output Explanation**

Print an integer value representing the player's score. If no balloons are shot then print 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 9 | 8 | 4 | 0 | 10 |
| **Output** | 54 | 33 | 4 | 0 | 88 |

**#Solution**

#include<stdio.h>

int fibo(int n)

{

int fibs[n];

fibs[0] = 0;

fibs[1] = 1;

int sum=0;

for(int i=2;i<n;i++)

{

fibs[i] = fibs[i-1]+fibs[i-2];

}

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

{

sum += fibs[i];

}

return sum;

}

int main()

{

int numBalloons;

scanf("%d",&numBalloons);

if (numBalloons==0)

{

printf("0");

}

else

{

int result = fibo(numBalloons);

printf("%d",result);

}

}

Q.15. Encryption key

The IT giant "SoftCompInfo" has decided to transfer its message through the network using a new encryption technique. The company has decided to encrypt the data using the non-prime number concept. The message is in the form of a number and the sum of non-prime digits present in the message is used as the encryption key. Write a function to determine the encryption key.

Example

Input

45673

Output

10

Explanation

The non-prime digits are 4 and 6. Hence the output is 4+6 = 10.

**Sample Input 1**

33512

**Sample Output 1**

0

**Sample Input 2**

468

**Sample Output 2**

18

**Input Explanation**

The input consists of a single integer value

**Output Explanation**

Output consists single integer value

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 6461 | 1001 | 33512 | 23456 | 48412 |
| **Output** | 16 | 0 | 0 | 10 | 12 |

**#Solution**

#include <stdio.h>

int nonprime(int);

int nonprime(int n)

{

int digit,i,sum=0,m;

while(n!=0)

{

digit=n%10;

n=n/10;

for(i=2;i<digit;i++)

{

if(digit%i==0)

{

sum=sum+digit;

break;

}

}

}

return sum;

}

int main()

{

int n;

scanf("%d",&n);

printf("%d",nonprime(n));

return 0;

}

Q.16. Perfect cube.

The children's toy-making company "ToysFun" is building cubic-shaped learning toys. The company has a list of N dimensions suggested by its designers but they wish to choose only those dimensions for the toys that are perfect cube numbers. To do this, they need to know the total count of perfect cube numbers present in the list of dimensions.

Write a function to help the toy manufacturers find the total count of perfect cube numbers present in the list of dimensions.

Example

Input

9

23 1 8 56 27 67 64 125 232

Output

5

Explanation

The cube numbers are 1, 8, 27, 64, 125. Hence the output is 5.

**Sample Input 1**

5

23 1 8

**Sample Output 1**

3

**Sample Input 2**

3

54 96 81

**Sample Output 2**

0

**Input Explanation**

The first line of input consists of an single integer, representing the total number of dimensions selected by the designers (N).

The second line of input consists of N space-separated integers - dimens1, dimens2, ...... dimensN-1 representing the value of the dimensions selected by the designers.

**Output Explanation**

Print an integer value representing the total count of the dimensions that are perfect cube numbers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** | 3  27 67 64 | 4  8 9 27 2 | 5  23 1 8 | 3  4 9 12 15 | 5  1 27 8 125 512 |
| **Output** | 2 | 3 | 3 | 0 | 5 |

**#Solution**

#include<stdio.h>

int dimens\_is\_cube(int n)

{

int i=1;

int result = 0;

while(i<=n)

{

if(i\*i\*i == n)

{

result = 1;

break;

}

i++;

}

return result;

}

int main()

{

int numDimensions;

scanf("%d", &numDimensions);

int dimens[numDimensions];

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

{

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

}

int perfect\_cube = 0;

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

{

if(dimens\_is\_cube(dimens[i]))

{

perfect\_cube++;

}

}

printf("%d",perfect\_cube);

return 0;

}

Q.17.

**Sample Input 1**

**Sample Output 1**

**Sample Input 2**

**Sample Output 2**

**Input Explanation**

**Output Explanation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** |  |  |  |  |  |
| **Output** |  |  |  |  |  |

**#Solution**

Q.18.

**Sample Input 1**

**Sample Output 1**

**Sample Input 2**

**Sample Output 2**

**Input Explanation**

**Output Explanation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** | **Test Case 4** | **Test Case 5** |
| **Input** |  |  |  |  |  |
| **Output** |  |  |  |  |  |

**#Solution**