TCS PROGRAMMING

Problem 1 :

Consider the below series:
1,2,1,3,2,5,3,7,5,11,8,13,13,17,

This series is a mixture of 2 series fail the odd terms in this series form a Fibonacci series and all the even terms are the prime numbers in ascending order

Write a program to find the Nth term in this series

The value N in a positive integer that should be read from mm. The Nth term that is calculated by the program should be written to STDOUT Other than the value of Nth term , no other characters / string or message should be written to STDOUT.

For example, when N:14, the 14th term in the series is 17 So only the value 17 should be printed to STDOUT

```
#include
void fibo(int);
void prime(int);
main()
{
int n,e;
scanf("%d",&n);

e=n/2;

if(n%2==0)
prime(e);


else
fibo(e+1);

}


void prime(int n)
{
int i,j,no,flag=0,count=0;
  for(i=1;i<=100;i++)
   {
flag=0;
       for(j=2;j<=i/2;j++)
       {
if(i%j==0)
flag=0;

else
flag=1;
        }

   if(flag==1)
    count++;

    if(count==n)
       {
          printf("%d\n",i);
          break;
       }
   }
}

void fibo(int n)
{
```

```
  int n0=0,n1=1,n2,i;
for(i=3;i<=n;i++)
{
    n2=n0+n1;
    n0=n1;
    n1=n2;
}
printf("%d",n2);
}
```

Or

```
#include
int main ()
{
int n, flag;
int i, t = 1, j;
int a, b, c, r, g;
a = 0;
b = 1;
c = 0;
r = 1;
scanf (“%d”, &n);
if (n % 2 != 0)
{
while (r <= (n / 2 + 1))
{
g = b;
c = a;
a = b;
b = a + c;
r++;
}
printf ("%d ", g);
}
else
{ //for prime no
for (i = 2; t <= (n / 2); i++)
{
flag = 0;
for (j = 2; j < i; j++)
{
if (i % j == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
{
t++;
}
}
printf ("%d ", i – 1);
}
return 0;
}
```

Problem 2:

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where **n** is the number of items.'

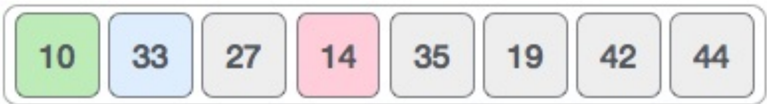
Consider the following depicted array as an example.

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

Click edit button to change this text. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut elit tellus, luctus nec ullamcorper mattis, pulvinar dapibus leo.

Click edit button to change this text. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut elit tellus, luctus nec ullamcorper mattis, pulvinar dapibus leo.

Click edit button to change this text. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut elit tellus, luctus nec ullamcorper mattis, pulvinar dapibus leo.



After two iterations, two least values are positioned at the beginning in a sorted manner.

The same process is applied to the rest of the items in the array.

Following is a pictorial depiction of the entire sorting process –

C

```
// C program for implementation of selection sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
int temp = *xp;
*xp = *yp;
*yp = temp;
}

void selectionSort(int arr[], int n)
{
int i, j, min_idx;

// One by one move boundary of unsorted subarray
for (i = 0; i < n-1; i++)
{
// Find the minimum element in unsorted array
min_idx = i;
```

```
for (j = i+1; j < n; j++)
if (arr[j] < arr[min_idx])
min_idx = j;

// Swap the found minimum element with the first element
swap(&arr[min_idx], &arr[i]);
}
}

/* Function to print an array */
void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", arr[i]);
printf("\n");
}

// Driver program to test above functions
int main()
{
int arr[] = {64, 25, 12, 22, 11};
int n = sizeof(arr)/sizeof(arr[0]);
selectionSort(arr, n);
printf("Sorted array: \n");
printArray(arr, n);
return 0;
}
```

Python

```
# Python program for implementation of Selection
# Sort
import sys
A = [64, 25, 12, 22, 11]

# Traverse through all array elements
for i in range(len(A)):

# Find the minimum element in remaining
# unsorted array
min_idx = i
for j in range(i+1, len(A)):
if A[min_idx] > A[j]:
min_idx = j

# Swap the found minimum element with
# the first element
A[i], A[min_idx] = A[min_idx], A[i]

# Driver code to test above
print ("Sorted array")
for i in range(len(A)):
print("%d" %A[i]),
```

Java

```
// Java program for implementation of Selection Sort
class SelectionSort
{
void sort(int arr[])
{
int n = arr.length;

// One by one move boundary of unsorted subarray
for (int i = 0; i < n-1; i++)
{
```

```
// Find the minimum element in unsorted array
int min_idx = i;
for (int j = i+1; j < n; j++)
if (arr[j] < arr[min_idx])
min_idx = j;

// Swap the found minimum element with the first
// element
int temp = arr[min_idx];
arr[min_idx] = arr[i];
arr[i] = temp;
}
}

// Prints the array
void printArray(int arr[])
{
int n = arr.length;
for (int i=0; i<n; ++i)
System.out.print(arr[i]+" ");
System.out.println();
}

// Driver code to test above
public static void main(String args[])
{
SelectionSort ob = new SelectionSort();
int arr[] = {64,25,12,22,11};
ob.sort(arr);
System.out.println("Sorted array");
ob.printArray(arr);
}
}
```

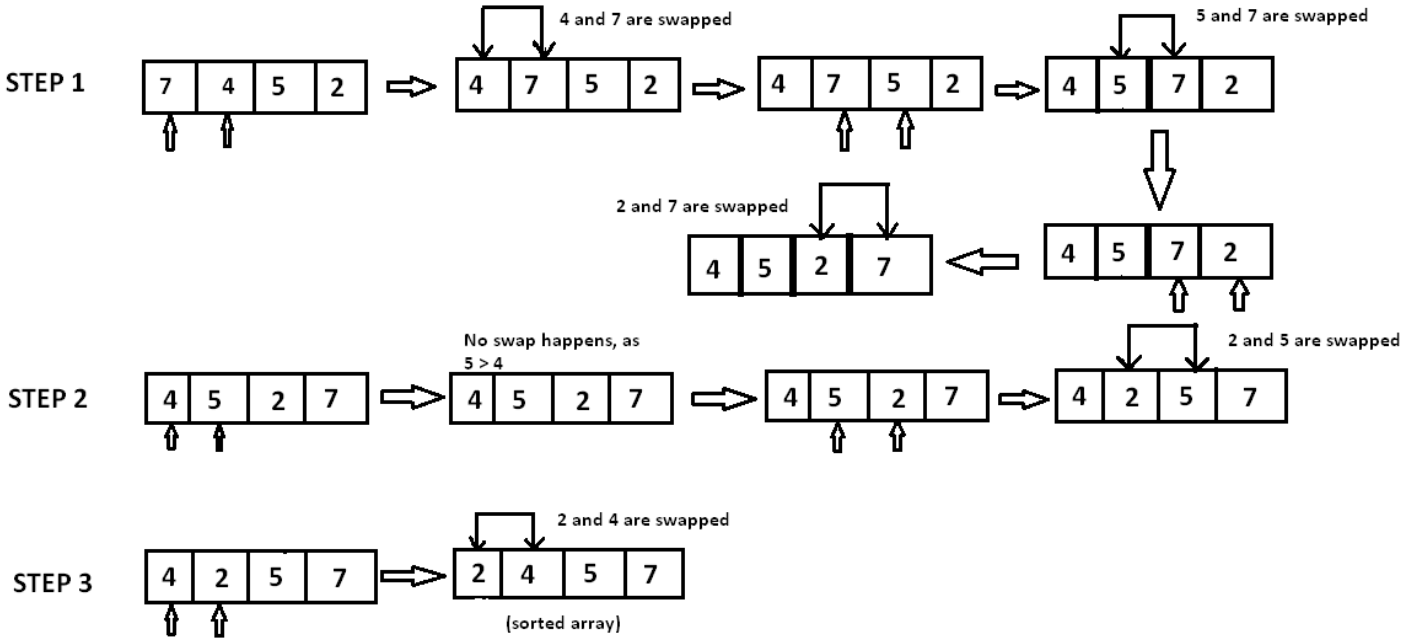
Problem 3 :

Bubble Sort

Sorting Algorithms are concepts that every competitive programmer must know. Sorting algorithms can be used for collections of numbers, strings, characters, or a structure of any of these types.

Bubble sort is based on the idea of repeatedly comparing pairs of adjacent elements and then swapping their positions if they exist in the wrong order.

Assume that A [] is an unsorted array of n elements. This array needs to be sorted in ascending order. The pseudo code is as follows:



C/C++

```
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
int temp = *xp;
*xp = *yp;
*yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)

// Last i elements are already in place
for (j = 0; j < n-i-1; j++)
if (arr[j] > arr[j+1])
swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
```

```
printf("%d ", arr[i]);
printf("n");
}

// Driver program to test above functions
int main()
{
int arr[] = {64, 34, 25, 12, 22, 11, 90};
int n = sizeof(arr)/sizeof(arr[0]);
bubbleSort(arr, n);
printf("Sorted array: \n");
printArray(arr, n);
return 0;
}
```

Java

```
// Java program for implementation of Bubble Sort
class BubbleSort
{
void bubbleSort(int arr[])
{
int n = arr.length;
for (int i = 0; i < n-1; i++)
for (int j = 0; j < n-i-1; j++)
if (arr[j] > arr[j+1])
{
// swap temp and arr[i]
int temp = arr[j];
arr[j] = arr[j+1];
arr[j+1] = temp;
}
}

/* Prints the array */
void printArray(int arr[])
{
int n = arr.length;
for (int i=0; i<n; ++i)
System.out.print(arr[i] + " ");
System.out.println();
}

// Driver method to test above
public static void main(String args[])
{
BubbleSort ob = new BubbleSort();
int arr[] = {64, 34, 25, 12, 22, 11, 90};
ob.bubbleSort(arr);
System.out.println("Sorted array");
ob.printArray(arr);
}
}
```

Python

```
# Python program for implementation of Bubble Sort

def bubbleSort(arr):
n = len(arr)

# Traverse through all array elements
for i in range(n):

# Last i elements are already in place
```

```
for j in range(0, n-i-1):

# traverse the array from 0 to n-i-1
# Swap if the element found is greater
# than the next element
if arr[j] > arr[j+1] :
arr[j], arr[j+1] = arr[j+1], arr[j]

# Driver code to test above
arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print ("Sorted array is:")
for i in range(len(arr)):
print ("%d" %arr[i]),
```

Problem 4 : Command Line Programming

Longest Common Subsequence –

LCS Problem Statement*:* Given two sequences, find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous.

For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”. So a string of length n has 2^n different possible subsequences.

It is a classic computer science problem, the basis of diff (a file comparison program that outputs the differences between two files), and has applications in bioinformatics.

Examples:
LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.
LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

C/C++

```
/* A Naive recursive implementation of LCS problem */
#include<bits/stdc++.h>

int max(int a, int b);

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
int lcs( char *X, char *Y, int m, int n )
{
if (m == 0 || n == 0)
return 0;
if (X[m-1] == Y[n-1])
return 1 + lcs(X, Y, m-1, n-1);
else
return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
}

/* Utility function to get max of 2 integers */
int max(int a, int b)
{
return (a > b)? a : b;
}

/* Driver program to test above function */
int main()
{
char X[] = “AGGTAB”;
char Y[] = “GXTXAYB”;

int m = strlen(X);
int n = strlen(Y);

printf(“Length of LCS is %d”, lcs( X, Y, m, n ) );

return 0;
}
```

Java

```
/* A Naive recursive implementation of LCS problem in java*/
public class LongestCommonSubsequence
{

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
int lcs( char[] X, char[] Y, int m, int n )
```

```
{
if (m == 0 || n == 0)
return 0;
if (X[m-1] == Y[n-1])
return 1 + lcs(X, Y, m-1, n-1);
else
return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
}

/* Utility function to get max of 2 integers */
int max(int a, int b)
{
return (a > b)? a : b;
}

public static void main(String[] args)
{
LongestCommonSubsequence lcs = new LongestCommonSubsequence();
String s1 = "AGGTAB";
String s2 = "GXTXAYB";

char[] X=s1.toCharArray();
char[] Y=s2.toCharArray();
int m = X.length;
int n = Y.length;

System.out.println("Length of LCS is" + " " +
lcs.lcs( X, Y, m, n ) );
}

}
```

Python

```
# A Naive recursive Python implementation of LCS problem

def lcs(X, Y, m, n):

if m == 0 or n == 0:
return 0;
elif X[m-1] == Y[n-1]:
return 1 + lcs(X, Y, m-1, n-1);
else:
return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));


# Driver program to test the above function
X = "AGGTAB"
Y = "GXTXAYB"
print "Length of LCS is ", lcs(X , Y, len(X), len(Y))
```

Problem 5

SubString Problem in C

Given a string as an input. We need to write a program that will print all non-empty substrings of that given string.

C++

```
/ C++ program to print all possible
// substrings of a given string

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

// Function to print all sub strings
void subString(char str[], int n)
{
// Pick starting point
for (int len = 1; len <= n; len++)
{
// Pick ending point
for (int i = 0; i <= n – len; i++)
{
// Print characters from current
// starting point to current ending
// point.
int j = i + len – 1;
for (int k = i; k <= j; k++)
cout << str[k];

cout << endl;
}
}
}

// Driver program to test above function
int main()
{
char str[] = “abc”;
subString(str, strlen(str));
return 0;
}
```

Java

```
// Java program to print all substrings of a string
public class GFG {

// Function to print all substring
public static void SubString(String str, int n)
{
for (int i = 0; i < n; i++)
for (int j = i+1; j <= n; j++)

// Please refer below article for details
// of substr in Java
// https://www.geeksforgeeks.org/java-lang-string-substring-java/
System.out.println(str.substring(i, j));
}

public static void main(String[] args)
{
String str = “abcd”;
SubString(str, str.length());
}
}
```

Problem 6 :

Pythrogorous Triplets

A Pythagorean triplet is a set of three integers a, b and c such that $a^2 + b^2 = c^2$. Given a limit, generate all Pythagorean Triples with values smaller than given limit.

```
Input : limit = 20

Output : 3 4 5
         8 6 10
         5 12 13
         15 8 17
         12 16 20
```

A **Simple Solution** is to generate these triplets smaller than given limit using three nested loop. For every triplet, check if Pythagorean condition is true, if true, then print the triplet. Time complexity of this solution is $O(limit^3)$ where 'limit' is given limit.

An **Efficient Solution** can print all triplets in O(k) time where k is number of triplets printed. The idea is to use square sum relation of Pythagorean triplet, i.e., addition of squares of a and b is equal to square of c, we can write these number in terms of m and n such that,

$$a = m^2 - n^2$$
$$b = 2 * m * n$$
$$c = m^2 + n^2$$

because,

$$a^2 = m^4 + n^4 - 2 * m^2 * n^2$$
$$b^2 = 4 * m^2 * n^2$$
$$c^2 = m^4 + n^4 + 2* m^2 * n^2$$

We can see that $a^2 + b^2 = c^2$, so instead of iterating for a, b and c we can iterate for m and n and can generate these triplets.

Below is the implementation of above idea :

C++

```
// C++ program to generate pythagorean
// triplets smaller than a given limit
#include <bits/stdc++.h>

// Function to generate pythagorean
// triplets smaller than limit
void pythagoreanTriplets(int limit)
{

// triplet: a^2 + b^2 = c^2
int a, b, c = 0;

// loop from 2 to max_limitit
int m = 2;

// Limiting c would limit
// all a, b and c
while (c < limit) {

// now loop on j from 1 to i-1
for (int n = 1; n < m; ++n) {

// Evaluate and print triplets using
// the relation between a, b and c
a = m * m - n * n;
b = 2 * m * n;
```

```
c = m * m + n * n;

if (c > limit)
break;

printf("%d %d %d\n", a, b, c);
}
m++;
}
}

// Driver Code
int main()
{
int limit = 20;
pythagoreanTriplets(limit);
return 0;
}
```

Java

```
// Java program to generate pythagorean
// triplets smaller than a given limit
import java.io.*;
import java.util.*;

class GFG {

// Function to generate pythagorean
// triplets smaller than limit
static void pythagoreanTriplets(int limit)
{

// triplet: a^2 + b^2 = c^2
int a, b, c = 0;

// loop from 2 to max_limitit
int m = 2;

// Limiting c would limit
// all a, b and c
while (c < limit) {

// now loop on j from 1 to i-1
for (int n = 1; n < m; ++n) {
// Evaluate and print
// triplets using
// the relation between
// a, b and c
a = m * m - n * n;
b = 2 * m * n;
c = m * m + n * n;

if (c > limit)
break;

System.out.println(a + " " + b + " " + c);
}
m++;
}
}

// Driver Code
public static void main(String args[])
{
int limit = 20;
pythagoreanTriplets(limit);
```

```
}
}
```

Python

```
# Python3 program to generate pythagorean
# triplets smaller than a given limit

# Function to generate pythagorean
# triplets smaller than limit
def pythagoreanTriplets(limits) :
c, m = 0, 2

# Limiting c would limit
# all a, b and c
while c < limits :

# Now loop on n from 1 to m-1
for n in range(1, m) :
a = m * m - n * n
b = 2 * m * n
c = m * m + n * n

# if c is greater than
# limit then break it
if c > limits :
break

print(a, b, c)

m = m + 1


# Driver Code
if __name__ == '__main__' :

limit = 20
pythagoreanTriplets(limit)
```

Problem 7 :

Armstrong Number
Given a number x, determine whether the given number is Armstrong number or not. A positive integer of **n digits** is called an Armstrong number of **order n** (order is number of digits) if.

```
abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ....
```

Example:

```
Input : 153
Output : Yes
153 is an Armstrong number.
1*1*1 + 5*5*5 + 3*3*3 = 153

Input : 120
Output : No
120 is not a Armstrong number.
1*1*1 + 2*2*2 + 0*0*0 = 9

Input : 1253
Output : No
1253 is not a Armstrong Number
1*1*1*1 + 2*2*2*2 + 5*5*5*5 + 3*3*3*3 = 723

Input : 1634
Output : Yes
1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4 = 1634
```

C/C++

```
// C++ program to determine whether the number is
// Armstrong number or not
#include<bits/stdc++.h>
using namespace std;

/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
if( y == 0)
return 1;
if (y%2 == 0)
return power(x, y/2)*power(x, y/2);
return x*power(x, y/2)*power(x, y/2);
}

/* Function to calculate order of the number */
```

```
int order(int x)
{
int n = 0;
while (x)
{
n++;
x = x/10;
}
return n;
}

// Function to check whether the given number is
// Armstrong number or not
bool isArmstrong(int x)
{
// Calling order function
int n = order(x);
int temp = x, sum = 0;
while (temp)
{
int r = temp%10;
sum += power(r, n);
temp = temp/10;
}

// If satisfies Armstrong condition
return (sum == x);
}

// Driver Program
int main()
{
int x = 153;
cout << isArmstrong(x) << endl;
x = 1253;
cout << isArmstrong(x) << endl;
return 0;
}
```

Java

```
// Java program to determine whether the number is
// Armstrong number or not
public class Armstrong
{
/* Function to calculate x raised to the
power y */
int power(int x, long y)
{
if( y == 0)
return 1;
if (y%2 == 0)
return power(x, y/2)*power(x, y/2);
return x*power(x, y/2)*power(x, y/2);
}

/* Function to calculate order of the number */
int order(int x)
{
int n = 0;
while (x != 0)
{
n++;
x = x/10;
}
return n;
}

// Function to check whether the given number is
// Armstrong number or not
boolean isArmstrong (int x)
{
// Calling order function
int n = order(x);
int temp=x, sum=0;
while (temp!=0)
{
int r = temp%10;
sum = sum + power(r,n);
temp = temp/10;
}

// If satisfies Armstrong condition
return (sum == x);
}

// Driver Program
public static void main(String[] args)
{
Armstrong ob = new Armstrong();
int x = 153;
System.out.println(ob.isArmstrong(x));
x = 1253;
System.out.println(ob.isArmstrong(x));
}
}
```

Python

```
# Python program to determine whether the number is
# Armstrong number or not

# Function to calculate x raised to the power y
def power(x, y):
if y==0:
return 1
if y%2==0:
return power(x, y/2)*power(x, y/2)
return x*power(x, y/2)*power(x, y/2)

# Function to calculate order of the number
def order(x):

# variable to store of the number
n = 0
while (x!=0):
n = n+1
x = x/10
return n

# Function to check whether the given number is
# Armstrong number or not
def isArmstrong (x):
n = order(x)
temp = x
sum1 = 0
while (temp!=0):
r = temp%10
sum1 = sum1 + power(r, n)
temp = temp/10

# If condition satisfies
return (sum1 == x)


# Driver Program
x = 153
print(isArmstrong(x))
x = 1253
print(isArmstrong(x))
```

Problem 8:

The square root of a Prime number by checking first if it is a prime number?

Write a C program which will check whether a given number N is a Prime or Not. If the Number N is a Prime, then find it's square root and print that value to the STDOUT as floating point number with exactly 2 decimal precision.
If the number is not Prime, then print the value 0.00 to STDOUT.
The given number will be positive non zero integer and it will be passed to the program as first command line argument.
Other than floating point No other information should be printed to STDOUT.

Solution:

[code language="cpp"]

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<math.h>
bool isPrime(int n)
{
if(n<2)
return false;
int i;
for(i=2;i*i<=n;i++)
{
if(n%i==0)
return false;
}
return true;
}
int main(int argc, char *argv[])
{
if(argc==1)
{
printf("No arguments");
return 0;
}
else
{
int n;
n=atoi(argv[1]);
float sq=0;
if(isPrime(n))
{
sq=sqrt(n);
printf("%.2f",sq);
}
else
printf("%.2f",sq);
return 0;
}
}
```

[/code]

Problem 9 :

Basic Program for Decimal to Octal

Given a decimal number as input, we need to write a program to convert the given decimal number into equivalent octal number. i.e convert the number with base value 10 to base value 8. The base value of a number system determines the number of digits used to represent a numeric value. For example, the binary number system uses two digits 0 and 1, octal number system uses 8 digits from 0-7 and decimal number system uses 10 digits 0-9 to represent any numeric value.

Examples:

```
Input : 16
Output : 20


Input : 10
Output : 12


Input: 33
Output: 41
```

Algorithm:

1. Store the remainder when the number is divided by 8 in an array.
2. Divide the number by 8 now
3. Repeat the above two steps until the number is not equal to 0.
4. Print the array in reverse order now.

For Example:
If the given decimal number is 16.
Step 1: Remainder when 16 is divided by 8 is 0. Therefore, arr[0] = 0.
Step 2: Divide 16 by 8. New number is 16/8 = 2.
Step 3: Remainder when 2 is divided by 8 is 2. Therefore, arr[1] = 2.
Step 4: Divide 2 by 8. New number is 2/8 = 0.
Step 5: Since number becomes = 0. Stop repeating steps and print the array in reverse order. Therefore the equivalent octal number is 20.

C/C++

```
// C++ program to convert a decimal
// number to octal number

#include <iostream>
using namespace std;

// function to convert decimal to octal
void decToOctal(int n)
{

// array to store octal number
int octalNum[100];

// counter for octal number array
int i = 0;
while (n != 0) {

// storing remainder in octal array
```

```
octalNum[i] = n % 8;
n = n / 8;
i++;
}

// printing octal number array in reverse order
for (int j = i - 1; j >= 0; j--)
cout << octalNum[j];
}

// Driver program to test above function
int main()
{
int n = 33;

decToOctal(n);

return 0;
}
```

Java

```
// Java program to convert a decimal
// number to octal number
import java.io.*;

class GFG
{
// Function to convert decimal to octal
static void decToOctal(int n)
{
// array to store octal number
int[] octalNum = new int[100];

// counter for octal number array
int i = 0;
while (n != 0)
{
// storing remainder in octal array
octalNum[i] = n % 8;
n = n / 8;
i++;
}

// Printing octal number array in reverse order
for (int j = i - 1; j >= 0; j--)
System.out.print(octalNum[j]);
}

// driver program
public static void main (String[] args)
{
int n = 33;
decToOctal(n);
}
}
```

Command Line Program to Convert Decimal to Octal

This is very smart short and quick program –

```
#include
int main(int argc,char *argv[])
{
        int n,s=0,b=1,r;
        n=atoi(argv[1]);
        int c=n;
        while(c>0)
        {
                r=c%8;
                s=s+r*b;
                c=c/8;
                b=b*10;
        }
        printf("%d",s);
        getch();
}
```

Problem 10

Basic Program to Binary to Octal

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

int convertBinarytoOctal(long long binaryNumber);
int main()
{
    long long binaryNumber;

    printf("Enter a binary number: ");
    scanf("%lld", &binaryNumber);

    printf("%lld in binary = %d in octal", binaryNumber, convertBinarytoOctal(binaryNumber)
);

    return 0;
}

int convertBinarytoOctal(long long binaryNumber)
{
    int octalNumber = 0, decimalNumber = 0, i = 0;

    while(binaryNumber != 0)
    {
        decimalNumber += (binaryNumber%10) * pow(2,i);
        ++i;
        binaryNumber/=10;
    }

    i = 1;

    while (decimalNumber != 0)
    {
        octalNumber += (decimalNumber % 8) * i;
        decimalNumber /= 8;
        i *= 10;
    }

    return octalNumber;
}
```

Output

```
Enter a binary number: 101001
101001 in binary = 51 in octal
import java.util.*;

public class Exercise24 {
public static void main(String[] args)

    {
        int binnum, binnum1,rem, decnum=0, quot, i=1, j;
        int octnum[] = new int[100];

        Scanner scan = new Scanner(System.in);

               System.out.print("Input a Binary Number : ");
        binnum = scan.nextInt();
        binnum1=binnum;

      while(binnum > 0)
```

```
        {
            rem = binnum % 10;
            decnum = decnum + rem*i;
            //System.out.println(rem);
            i = i*2;
            binnum = binnum/10;
        }

                i=1;
        quot = decnum;

        while(quot > 0)
        {
            octnum[i++] = quot % 8;
            quot = quot / 8;
        }

        System.out.print("Equivalent Octal Value of " +binnum1+ " is :");
        for(j=i-1; j>0; j--)
        {
            System.out.print(octnum[j]);
        }
 System.out.print("\n");

    }
}
```

Sample Output:

```
Enter Binary Number : 111

Equivalent Octal Value of 111 is :7
```

Command Line Program to Convert Binary to Octal

This is a very smart program very short and quick method –

```
#include
void main(int argc,char *argv[])

{
   long int n,r,c,b=1,s=0;
    n=atoi(argv[1]);
    c=n;
    while(c!=0)
    {
    r=c%10;
    s=s+r*b;
    c=c/10;
    b=b*2;
        }
        printf("%lo",s);
        getch();
}
```

Problem 11:
To check if a year is Leap year or not

C/C++

```
#include <stdio.h>

int main()
{
int year;

printf("Enter a year: ");
scanf("%d",&year);

if(year%4 == 0)
{
if( year%100 == 0)
{
// year is divisible by 400, hence the year is a leap year
if ( year%400 == 0)
printf("%d is a leap year.", year);
else
printf("%d is not a leap year.", year);
}
else
printf("%d is a leap year.", year );
}
else
printf("%d is not a leap year.", year);

return 0;
}
```

Java

```
import java.util.Scanner;
public class Check_Leap_Year
{
public static void main(String args[])
{
Scanner s = new Scanner(System.in);
System.out.print("Enter any year:");
int year = s.nextInt();
boolean flag = false;
if(year % 400 == 0)
{
flag = true;
}
else if (year % 100 == 0)
{
flag = false;
}
else if(year % 4 == 0)
{
flag = true;
}
else
{
flag = false;
}
if(flag)
{
System.out.println("Year "+year+" is a Leap Year");
```

```
}
else
{
System.out.println("Year "+year+" is not a Leap Year");
}
}
}
```

Command Line

```
#include
void main(int argc,char *argv[])
{
int n;
n=atoi(argv[1]);
if(n%4==0)
{
if(n%100==0)
{
if(n%400==0)
printf("Leap Year");
else printf("Not Leap Year");
}
else printf("Leap Year");
}
else
printf("Not Leap Year");
getch(); }
```

Problem 12:
Command Line Program to Check if a Number is Prime or Not

C/C++

```
#include <stdio.h>
int main()
{
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2; i<=n/2; ++i)
    {
   // condition for nonprime number
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }

    if (flag==0)
        printf("%d is a prime number.",n);
    else
        printf("%d is not a prime number.",n);

    return 0;
}
```

Java

```
public class Prime {

    public static void main(String[] args) {

        int num = 29;
        boolean flag = false;
        for(int i = 2; i <= num/2; ++i)
        {
            // condition for nonprime number
            if(num % i == 0)
            {
                flag = true;
                break;
            }
        }

        if (!flag)
            System.out.println(num + " is a prime number.");
        else
            System.out.println(num + " is not a prime number.");
    }
}
```

Command Line

```
#include
int main(int argc, char *argv[])

{
             int n, i, flag = 0;
             n = atol(argv[1]);
            for(i=2; i<=n/2; ++i)

   {
            if(n%i==0)

      {
            flag=1;
            break;
      }
   }
   if (flag==0)
            printf("%d is a prime number.",n);
   else
            printf("%d is not a prime number.",n);
   return 0;
}
```

Problem 13
Command Line Program to Reverse a Number

C/C++

```
#include <stdio.h>
int main()
{
    int n, reversedNumber = 0, remainder;

    printf("Enter an integer: ");
    scanf("%d", &n);

    while(n != 0)
    {
        remainder = n%10;
        reversedNumber = reversedNumber*10 + remainder;
        n /= 10;
    }
    printf("Reversed Number = %d", reversedNumber);

    return 0;
}
```

Java

```
public class ReverseNumber {
    public static void main(String[] args) {
        int num = 1234, reversed = 0;
        while(num != 0) {
            int digit = num % 10;
            reversed = reversed * 10 + digit;
            num /= 10;
        }
        System.out.println("Reversed Number: " + reversed);
    }
}
```

Command Line Programming

```
#include
#include
int main(int argc, char *argv[])
{
if(argc==1)
{
printf("No Arguments");
return 0;
}
else
{
int n,reverseNumber,temp,rem;
n=atoi(argv[1]);
temp=n;
reverseNumber=0;
while(temp)
{
rem=temp%10;
reverseNumber=reverseNumber*10+rem;
```

```
temp=temp/10;
}
printf("%d",reverseNumber);
return 0;
}
}
```

Problem 14

Reverse without in built Functions

C/C++

```
#include <stdio.h>
int main()
{
   char s[1000], r[1000];
   int begin, end, count = 0;

   printf("Input a string\n");
   gets(s);

   // Calculating string length

   while (s[count] != '\0')
      count++;

   end = count - 1;

   for (begin = 0; begin < count; begin++) {
      r[begin] = s[end];
      end--;
   }
   r[begin] = '\0';
   printf("%s\n", r);
   return 0;
}
```

Problem 15

GCD of three numbers

C

Definition of HCF (Highest common factor):

HFC is also called greatest common divisor (gcd). HCF of two numbers is a largest positive numbers which can divide both numbers without any remainder. For example HCF of two numbers 4 and 8 is 2 since 2 is the largest positive number which can dived 4 as well as 8 without a remainder.

Logic for writing program:

It is clear that any number is not divisible by greater than number itself.

☆In case of more than one numbers, a possible maximum number which can divide all of the numbers must be minimum of all of that numbers.

For example: 10, 20, and 30
Min (10, 20, 30) =10 can divide all there numbers. So we will take one for loop which will start form min of the numbers and will stop the loop when it became one, since all numbers are divisible by one. Inside for loop we will write one if conditions which will check divisibility of both the numbers.

Program :

```
#include<stdio.h>
int gcd(int , int , int);
int main()
{
int i , j , k , g;
scanf("%d %d %d", &i , &j , &k);

g = gcd(i , j , k);
printf("%d",g);

return 0;
}

int gcd(int i , int j , int k)
{
int least;
least = i;
while(!( (i == j) && (j == k) ) )
{
i = (i == 0 ? least : i);
j = (j == 0 ? least : j);
k = (k == 0 ? least : k);
if(i <= j)
{
if(i <= k)
least = i;
else
least = k;
}
else
{
if(j <= k)
least = j;
else
least = k;
}
i = i % least;
j = j % least;
k = k % least;
```

```
}
return least;

}
```

Java

```
import java.util.*;

public class HCFOFNumbers {
	public static int hcf(int a, int b) {
		if (b == 0)
			return a;
		else
			return hcf(b, a % b);
	}

	public static int hcf(int a, int b, int c) {

		return hcf(hcf(a, b), c);

	}

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.print("Enter Number 1: ");
		int num1 = input.nextInt();
		System.out.print("Enter Number 2: ");
		int num2 = input.nextInt();
		System.out.print("Enter Number 3: ");
		int num3 = input.nextInt();
		int hcfOfNumbers = HCFOFNumbers.hcf(num1, num2, num3);
		System.out.println("HCF of three numbers " + num1 + "," + num2
				+ " and " + num3 + " is: " + hcfOfNumbers);
	}
}
```

Problem 16:

Sum of odd number in given range using Command Line Programming

```
 [code language="cpp"]

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{

if(argc==1 || argc >3){
printf("Please enter only two arguemnts one lower limit and one upper limit");
return 0;
}

else {

int i, n,m, sum=0;
n=atoi(argv[1]);
m = atoi(argv[2]);
for(i=n; i<=m; i++)
{
if(i%2==1)
sum += i;
}

printf("Sum of odd numbers = %d", sum);
}
return 0;
}

[/code]
```

Problem 17:

Write a program that will take a number in string format and convert it and print it in integer format. For e.g.,

Input String => ”574”
Output Integer => 574

```
#include <stdio.h>

#include <string.h>

#include <math.h>

#include <ctype.h>

void read_string(char sample[]);

int check_string(char sample[]);

void convert_int(char sample[]);

int main()

{

int status;

char sample[100];

read_string(sample);

status = check_string(sample);

while ( status == 1 )

{

printf("\nYou are supposed to enter only digits[0-9]\n");

read_string(sample);

status = check_string(sample);

}

convert_int(sample);

return 0;

}
```

```
/*This function reads the number entered by the user and stores it in the character array sample. We are using fflush to flush the input buffer so that subsequent scans provide the desired input */

void read_string(char sample[])

{

printf(“\nEnter the string to be converted to integer(only numbers) :”);

strcpy(sample, “”);
```

```
gets(sample);
fflush(stdin);
}
/* This function ensures that the string consists of digits only. If it contains other characters such as punctuation characters, alphabets etc. then it returns 1 ( Error )
*/
int check_string(char sample[]) {
int i, flag = 0;
for ( i = 0; i<strlen(sample)-1; i++) {
if( isdigit( sample[i] )) {
flag = 1;
}
else {
flag = 0;
break;
}
}
if ( flag == 0 ) {
return 1;
}
else {
return 0;
}
}
/* This function does the conversion of the number in string format to integer format. A character when used in expressions is treated as an integer and the ASCII value of the character is used in the expression. For e.g., if the character is '4' then ASCII value of 4 is 0x34 which is decimal 52. If we want the integer 4 then we subtract the 48 from the ASCII value of 4 expressed in decimal form, i.e., 52 – 48 = 4
*/
/* pow function is used to calculate exponent and you have to include math.h and separately link the math library as shown ( Unix/Linux )
$ gcc convert.c -lm
*/
void convert_int(char sample[]) {
int i, sum = 0, j = 0;
for ( i=strlen(sample)-1; i>=0; i– ) {
sum = sum + (sample[i] – 48)*pow(10, j);
j++;
}
printf("The converted integer = %d\n", sum);
}
```

Problem 18:

Read from a File

Write a program to print the number of characters and lines in a file. You should ignore the space, tab and newline characters for the character count

Create a file called input.txt in the same directory as this file and enter some lines of text or copy a paragraph of text. The filename should be input.txt

```
#include <stdio.h>

#include <stdlib.h>

int main() {

FILE *fp;

int line_count = 0, char_count = 0;

char c;

fp = fopen("input.txt", "r");

if (fp == NULL) {

printf("\nThe file cannot be opened in read mode\n");

exit(1);

}

c = fgetc(fp);

while ( c != EOF ) {

/* If the character read is a newline or tab or space */

if ( c == '\n' || c == '\t' || c == '\b' ) {

/* If it is a newline character increment the line count */

if ( c == '\n')

line_count++;

/* Read the next character from the file */

c = fgetc(fp);

/* Continue with the next iteration */

continue;

}

/* Since the character is not a newline, tab or space increment

the character count

*/

char_count++;

c = fgetc(fp);

}
```

```
/* Since we are done processing the entire file we print the

the charcter count and the word count for the file

*/

printf("\n The number of characters = %d\n", char_count);

printf("\n The number of lines = %d\n", line_count);

return 0;

}
```

Problem 19:

Here we can take the string and reverse it. Then compare the original string with the reversed string. If they are the same then it is a palindrome.

Write a program that will take a number in integer format and convert it and print it in string format. For e.g.,

Input Integer => 574
Output String => "574"

(using sprintf)

```
/* sprint usage is just like printf. Instead of printing the value on the console we
print the value into a string ( char array ).
printf => Print on Console
fprintf => Print into a file
sprint => Print into a string
*/
#include <stdio.h>
#include <string.h>
int main()
{
int number, nitems;
char clear[25];
char token[25];
printf("\nPlease enter a number :");
fflush(stdin);
nitems = scanf("%d", &number);
while ( nitems != 1 ){
/* Clear the Buffer */
gets(clear);
printf("\nPlease enter a number – digits only :");
nitems = scanf("%d", &number);
}
printf("\nThe number of items scanned = %d", nitems);
sprintf(token, "%d", number);
printf("\nThe number %d is converted into string : %s\n", number, token);
}
```

Solution 2 (using integer manipulation)

```
#include <stdio.h>
#include <string.h>
int main() {
int number, temp, j = 0, i, rem;
char token[25], snumber[25];
printf("\nPlease enter a number :");
scanf("%d", &number);
```

```
temp = number;
while ( temp ) {
rem = temp%10;
temp = temp/10;
token[j++] = (char)(rem + 48);
}
token[j] = '\0'; /* Token Array has the value "427" */
i = 0;
/* Here we are reversing the array, i.e., 427 to 724 */
for ( j=strlen(token)-1; j>=0; j- ){
snumber[i++] = token[j];
}
snumber[i] = '\0';
printf("\nThe number %d is converted into string : %s\n", number,
snumber);
}
```

Problem 20

Write a program to convert a number to its binary equivalent and print the bit representation of the number

Solution (Using Bit Operators & Masking)

```
/* In this method we create a mask with the most significant bit set and And ( &) the
mask and the number. If the result is 0 then we know that the Most Significant Bit (MSB)
of the number is 0. If the result is 1 then the MSB of the number is 1. Then we right
shift the mask by 1 bit and repeat the process.

*/

#include<stdio.h>

void binary(unsigned int);

void main() {

unsigned int num;

printf("Enter Decimal Number : ");

scanf("%u",&num);

binary(num);

}

void binary(unsigned int num) {

unsigned int mask=0x80000000;

printf("\nThe Binary Number is : ");

while(mask > 0){

if((num & mask) == 0 )

printf("0");

else

printf("1");

mask = mask >> 1;

}

printf("\n");

}
```

Problem 21:

Write a program to swap 2 numbers without using the temporary variable

The idea is to get sum in one of the two given numbers. The numbers can then be swapped using the sum and subtraction from sum.

C/C++

```
#include <stdio.h>
int main()
{
int x = 10, y = 5;

// Code to swap 'x' and 'y'
x = x + y; // x now becomes 15
y = x – y; // y becomes 10
x = x – y; // x becomes 5

printf("After Swapping: x = %d, y = %d", x, y);
return 0;
}
```

Java

```
// Program to swap two numbers without
// using temporary variable
import java.*;
class PrepInsta {
public static void main(String a[])
{
int x = 10;
int y = 5;
x = x + y;
y = x – y;
x = x – y;
System.out.println("After swaping:"
+ " x = " + x + ", y = " + y);
}
}
```

Python

```
x = 10
y = 5

# Code to swap ‘x’ and ‘y’
# x now becomes 15
x = x + y

# y becomes 10
y = x – y

# x becomes 5
x = x – y
print(“After Swapping: x =”,x ,” y =”, y);
```

Problem 22:

Pushing all 0’s

Given an array of random numbers, Push all the zero’s of a given array to the end of the array. For example, if the given arrays is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. The order of all other elements should be same. Expected time complexity is O(n) and extra space is O(1).

Example:

```
Input :  arr[] = {1, 2, 0, 4, 3, 0, 5, 0};
Output : arr[] = {1, 2, 4, 3, 5, 0, 0};


Input : arr[]  = {1, 2, 0, 0, 0, 3, 6};
Output : arr[] = {1, 2, 3, 6, 0, 0, 0};
```

C/C++

```
// A C++ program to move all zeroes at the end of array
#include <iostream>
using namespace std;

// Function which pushes all zeros to end of an array.
void pushZerosToEnd(int arr[], int n)
{
int count = 0; // Count of non-zero elements

// Traverse the array. If element encountered is non-
// zero, then replace the element at index 'count'
// with this element
for (int i = 0; i < n; i++)
if (arr[i] != 0)
arr[count++] = arr[i]; // here count is
// incremented

// Now all non-zero elements have been shifted to
// front and 'count' is set as index of first 0.
// Make all elements 0 from count to end.
while (count < n)
```

```
arr[count++] = 0;
}

// Driver program to test above function
int main()
{
int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
int n = sizeof(arr) / sizeof(arr[0]);
pushZerosToEnd(arr, n);
cout << "Array after pushing all zeros to end of array :n";
for (int i = 0; i < n; i++)
cout << arr[i] << " ";
return 0;
}
```

Java

```
/* Java program to push zeroes to back of array */
import java.io.*;

class PushZero
{
// Function which pushes all zeros to end of an array.
static void pushZerosToEnd(int arr[], int n)
{
int count = 0; // Count of non-zero elements

// Traverse the array. If element encountered is
// non-zero, then replace the element at index 'count'
// with this element
for (int i = 0; i < n; i++)
if (arr[i] != 0)
arr[count++] = arr[i]; // here count is
// incremented

// Now all non-zero elements have been shifted to
// front and 'count' is set as index of first 0.
// Make all elements 0 from count to end.
while (count < n)
arr[count++] = 0;
}

/*Driver function to check for above functions*/
public static void main (String[] args)
{
int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
int n = arr.length;
pushZerosToEnd(arr, n);
System.out.println("Array after pushing zeros to the back: ");
for (int i=0; i<n; i++)
System.out.print(arr[i]+" ");
}
}
```

Problem 23:

Write a function that accepts 2 strings search string and pattern string and returns TRUE if the pattern string is found in the search string and FALSE if the pattern string is not found in the search string

```
#include
#define TRUE 1
#define FALSE 0
int search(char sentence[], char pattern[]);
int main(){
char sentence[1000];
char pattern[25];
int result;
printf("Please enter a paragraph not exceeding 1000 characters :");
gets(sentence);
printf("\n\n");
printf("Please enter the search string :");
gets(pattern);
result = search(sentence, pattern);
}

int search(char sentence[], char pattern[]){
char *p;
/* The library function strstr searches for the pattern string in the sentence string. If the pattern is found it returns a pointer to the index of the pattern in the sentence. If not it returns NULL
*/
p = strstr(sentence, pattern);
if ( p == NULL ){
printf("\nThe search string was not found in the sentence\n\n");
return FALSE;
}
else {
printf("\nThe search string was found in the sentence\n\n");
return TRUE;
}
}
```

Problem 24

K’th Smallest/Largest Element in Unsorted Array

Given an array and a number k where k is smaller than size of array, we need to find the k’th smallest element in the given array. It is given that ll array elements are distinct.

Examples:

```
Input: arr[] = {7, 10, 4, 3, 20, 15}
       k = 3
Output: 7


Input: arr[] = {7, 10, 4, 3, 20, 15}
       k = 4
Output: 10
```

C/C++

```
// Simple C++ program to find k'th smallest element
#include<iostream>
#include<algorithm>
using namespace std;

// Function to return k'th smallest element in a given array
int kthSmallest(int arr[], int n, int k)
{
// Sort the given array
sort(arr, arr+n);

// Return k'th element in the sorted array
return arr[k-1];
}

// Driver program to test above methods
int main()
{
int arr[] = {12, 3, 5, 7, 19};
int n = sizeof(arr)/sizeof(arr[0]), k = 2;
cout << "K'th smallest element is " << kthSmallest(arr, n, k);
return 0;
}
```

Java

```
// Java code for kth smallest element
// in an array
import java.util.Arrays;
import java.util.Collections;

class GFG
{
// Function to return k'th smallest
// element in a given array
public static int kthSmallest(Integer [] arr,
int k)
```

```
{
// Sort the given array
Arrays.sort(arr);

// Return k'th element in
// the sorted array
return arr[k-1];
}

// driver program
public static void main(String[] args)
{
Integer arr[] = new Integer[]{12, 3, 5, 7, 19};
int k = 2;
System.out.print( "K'th smallest element is "+
kthSmallest(arr, k) );
}
}
```

Problem 25:

GCD Array of Numbers

The GCD of three or more numbers equals the product of the prime factors common to all the numbers, but it can also be calculated by repeatedly taking the GCDs of pairs of numbers.

```
gcd(a, b, c) = gcd(a, gcd(b, c))
             = gcd(gcd(a, b), c)
             = gcd(gcd(a, c), b)
```

For an array of elements, we do following.

```
result = arr[0]
For i = 1 to n-1
   result = GCD(result, arr[i])
```

Below is the implementation of above idea.

C/C++

```
// C++ program to find GCD of two or
// more numbers
#include <bits/stdc++.h>
using namespace std;

// Function to return gcd of a and b
int gcd(int a, int b)
{
if (a == 0)
return b;
return gcd(b % a, a);
}

// Function to find gcd of array of
// numbers
int findGCD(int arr[], int n)
{
int result = arr[0];
for (int i = 1; i < n; i++)
result = gcd(arr[i], result);
return result;
}

// Driven code
int main()
{
int arr[] = { 2, 4, 6, 8, 16 };
int n = sizeof(arr) / sizeof(arr[0]);
cout << findGCD(arr, n) << endl;
return 0;
}
```

Java

```
// Java program to find GCD of two or
// more numbers

public class GCD {
// Function to return gcd of a and b
static int gcd(int a, int b)
{
if (a == 0)
return b;
return gcd(b % a, a);
}

// Function to find gcd of array of
// numbers
static int findGCD(int arr[], int n)
{
int result = arr[0];
for (int i = 1; i < n; i++)
result = gcd(arr[i], result);

return result;
}

public static void main(String[] args)
{
int arr[] = { 2, 4, 6, 8, 16 };
int n = arr.length;
System.out.println(findGCD(arr, n));
}
}
```

Problem 26:

Write a function that reads a sentence into a string and splits the sentence into words using library functions

```
#include <stdio.h>
#include <string.h>
int main() {
char sentence[100];
char *p;
printf("\nPlease enter a sentence :");
gets(sentence);
p = strtok(sentence, "\t ");
while ( p != NULL ){
printf("%s\n", p);
p = strtok(NULL, "\t");
}
return 0;
}
```

Problem 27:

Write a function that removes all duplicate spaces from a sentence. Your program should ensure that only one space exists between words

```
/* We can remove duplicates by checking each character and having a space count. But here we are doing it using the string library function strtok
*/
#include <stdio.h>
#include <string.h>
int main(){
char input[100];
char output[100];
char temp[100];
char *p;
printf("\nPlease enter a sentence :");
gets(input);
strcpy(temp, input);
strcpy(output,"");
p = strtok(temp, " ");
while ( p != NULL ){
strcat(output,p);
strcat(output," ");
printf("%s\n", p);
p = strtok(NULL, " ");
}
printf("%s\n", input);
printf("%s\n", output);
printf("%d\n", strlen(output));
return 0;
}
```

Problem 28:

Write a function that uses character handling library functions to determine the number of upper case, lower case, digits, spaces and punctuation characters in the specified text

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int main()
{
int length, i;
int alpha = 0, punct = 0, digit = 0, lower = 0, upper = 0,
spaces = 0, others = 0;
char buffer[150] = “English Language consists of letters
from A to Z and digits 1 to 9 which are
used to form : words, sentences and
paragraphs!!!”;
length = strlen(buffer);
for ( i=0; i<length; i++ )
{
if( isalpha( buffer[i] ))
{
alpha++;
if ( islower( buffer[i] ))
lower++;
else
upper++;
}
else if (isdigit(buffer[i] ))
{
digit++;
}
else if (isspace(buffer[i])){
spaces++;
}
else if (ispunct(buffer[i])){
punct++;
}
else
others++;
}
printf(“The number of lowercase letters = %d\n”, lower);
```

```
printf(“The number of uppercase letters = %d\n”, upper);
printf(“The number of spaces = %d\n”, spaces);
printf(“The number of punctuation characters = %d\n”, punct);
printf(“The number of alphabets = %d\n”, alpha);
printf(“The number of digits = %d\n”, digit);
}
```

Problem 29:
Program to Capitalise a String

C

```
#include <stdio.h>
void upper_string(char []);
int main()
{
   char string[100];

   printf("Enter a string to convert it into upper case\n");
   gets(string);

   upper_string(string);

   printf("The string in upper case: %s\n", string);
   return 0;
}

void upper_string(char s[]) {
   int c = 0;

   while (s[c] != '\0') {
      if (s[c] >= 'a' && s[c] <= 'z') {
         s[c] = s[c] - 32;
      }
      c++;
   }
}
```

using strUPR Method

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

int main()
{
   char string[1000];

   printf("Input a string to convert to upper case\n");
   gets(string);

   printf("The string in upper case: %s\n", strupr(string));

   return  0;
}
```

Java

```
import java.io.*;
public class Test {

   public static void main(String args[]) {
      String Str = new String("Welcome to PrepInsta.com");

      System.out.print("Return Value :" );
      System.out.println(Str.toUpperCase() );
   }
}
```

Problem 30:
Fibonacci Series

C

```c
#include <stdio.h>
int main()
{
    int i, n, t1 = 0, t2 = 1, nextTerm;

    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");

    for (i = 1; i <= n; ++i)
    {
        printf("%d, ", t1);
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }
    return 0;
}
```

Java

```java
public class Fibonacci {

    public static void main(String[] args) {

        int n = 10, t1 = 0, t2 = 1;
        System.out.print("First " + n + " terms: ");

        for (int i = 1; i <= n; ++i)
        {
            System.out.print(t1 + " + ");

            int sum = t1 + t2;
            t1 = t2;
            t2 = sum;
        }
    }
}
```

Problem 31:
Perfect Square or not Program

C

```
/*C program to check number is perfect square or not.*/

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

int main()
{
    int num;
    int iVar;
    float fVar;

    printf("Enter an integer number: ");
    scanf("%d",&num);

    fVar=sqrt((double)num);
    iVar=fVar;

    if(iVar==fVar)
        printf("%d is a perfect square.",num);
    else
        printf("%d is not a perfect square.",num);

    return 0;
}
```

Java

```
// Java program to find if x is a
// perfect square.
class PI {

static boolean isPerfectSquare(double x)
{

// Find floating point value of
// square root of x.
double sr = Math.sqrt(x);

// If square root is an integer
return ((sr - Math.floor(sr)) == 0);
}

// Driver code
public static void main(String[] args)
{
double x = 2500;

if (isPerfectSquare(x))
System.out.print("Yes");
else
System.out.print("No");
}
}
```

Problem 32:
Matrix Multiplication Program

C

```
#include <stdio.h>

int main()
{
  int m, n, p, q, c, d, k, sum = 0;
  int first[10][10], second[10][10], multiply[10][10];

  printf("Enter number of rows and columns of first matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter elements of first matrix\n");

  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      scanf("%d", &first[c][d]);

  printf("Enter number of rows and columns of second matrix\n");
  scanf("%d%d", &p, &q);

  if (n != p)
    printf("The matrices can't be multiplied with each other.\n");
  else
  {
    printf("Enter elements of second matrix\n");

    for (c = 0; c < p; c++)
      for (d = 0; d < q; d++)
        scanf("%d", &second[c][d]);

    for (c = 0; c < m; c++) {
      for (d = 0; d < q; d++) {
        for (k = 0; k < p; k++) {
          sum = sum + first[c][k]*second[k][d];
        }

        multiply[c][d] = sum;
        sum = 0;
      }
    }

    printf("Product of the matrices:\n");

    for (c = 0; c < m; c++) {
      for (d = 0; d < q; d++)
        printf("%d\t", multiply[c][d]);

      printf("\n");
    }
  }

  return 0;
}
```

Java

```
public class MultiplyMatrices {

    public static void main(String[] args) {
        int r1 = 2, c1 = 3;
        int r2 = 3, c2 = 2;
        int[][] firstMatrix = { {3, -2, 5}, {3, 0, 4} };
        int[][] secondMatrix = { {2, 3}, {-9, 0}, {0, 4} };

        // Mutliplying Two matrices
        int[][] product = new int[r1][c2];
        for(int i = 0; i < r1; i++) {
            for (int j = 0; j < c2; j++) {
                for (int k = 0; k < c1; k++) {
                    product[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
                }
            }
        }

        // Displaying the result
        System.out.println("Sum of two matrices is: ");
        for(int[] row : product) {
            for (int column : row) {
                System.out.print(column + "    ");
            }
            System.out.println();
        }
    }
}
```

Problem 33:
Program to Print all Permutations of a String

C

```
// C program to print all permutations with duplicates allowed
#include <stdio.h>
#include <string.h>

/* Function to swap values at two pointers */
void swap(char *x, char *y)
{
char temp;
temp = *x;
*x = *y;
*y = temp;
}

/* Function to print permutations of string
This function takes three parameters:
1. String
2. Starting index of the string
3. Ending index of the string. */
void permute(char *a, int l, int r)
{
int i;
if (l == r)
printf("%s\n", a);
else
{
for (i = l; i <= r; i++)
{
swap((a+l), (a+i));
permute(a, l+1, r);
swap((a+l), (a+i)); //backtrack
}
}
}

/* Driver program to test above functions */
int main()
{
char str[] = "ABC";
int n = strlen(str);
permute(str, 0, n-1);
return 0;
}
```

Java

```
package com.java2novice.algo;

import java.util.ArrayList;
import java.util.List;

public class StringPermutationsEx {

    public static void main(String a[]) {

        List<String> output = StringPermutationsEx.generatePermutations("xyz");
        System.out.println("Result size: "+output.size());
        output.stream().forEach(System.out::println);
        System.out.println("------------------");

        output = StringPermutationsEx.generatePermutations("ABCD");
```

```
        System.out.println("Result size: "+output.size());
        output.stream().forEach(System.out::println);
    }

    public static List<String> generatePermutations(String input) {

        List<String> strList = new ArrayList<String>();
        StringPermutationsEx.permutations("", input, strList);

        return strList;
    }

    private static void permutations(String consChars, String input, List<String> opContain
er) {

        if(input.isEmpty()) {
            opContainer.add(consChars+input);
            return;
        }

        for(int i=0; i<input.length(); i++) {
            permutations(consChars+input.charAt(i),
                         input.substring(0, i)+input.substring(i+1),
                         opContainer);
        }
    }
}
```

Problem 34:
Write a program to multiply a number by 8 without using the * operator
Just left shift by 3 bits. By left shifting a number by 1 bit we are multiplying by 2.
Left Shift by 1 bit => Multiply by 2
Left Shift by 2 bit => Multiply by 4
Left Shift by 3 bit => Multiply by 8
The number 8 => 00001000
Left Shift by 1 bit => 00010000 (16)
Left Shift by 2 bit => 00100000 (32)
Left Shift by 3 bit => 01000000 (34)

Problem 35:
Second Largest Number in an Array using Command Line Programming

Problem 36:
Addition of 2 numbers and printing the result in binary Using Command Line Programming

Problem 37:
Write a function to calculate the length of a string without using strlen function using Command Line Programming

Problem 38:
Write a function that reads a sentence into a string and splits the sentence into words without using library functions
You have to do this without using the strtok function. Since the delimiters are not specified you assume the default delimiters such as spaces, comma, tabs and newlines
You examine character by character and when you encounter a delimiter you have to terminate the word with the NULL character(\0)

Problem 39:
Write a function that takes an array called scores and 2 pointer parameters min and max and determines the minimum and maximum values and returns them to the calling function

Problem 40:
Write a Program to remove vowels from a string?

Problem 41:
Write a Program for Matrix Multiplication(Universal Program)

Problem 42:
Provide a fast way to multiply a number by 31
Left shift the number by 5 bits and subtract the number once. For e.g., If the number is 10 left shift by 5 bits gives.
Left Shift by 1 bit is multiplication by 2
Left Shift by 5 bits is multiplication by 32
Since we need to multiply by 31 what we do is multiply the number by 32 and subtract the number once.

Problem 43:

Consider the below series:
1, 2, 1, 3, 2, 5, 3, 7, 5, 11, 8, 13, 13, 17, …
This series is a mixture of 2 series – all the odd terms in this series form a Fibonacci series and all the even terms are the prime numbers in ascending order.
Write a program to find the Nth term in this series.
The value N is a Positive integer that should be read from STDIN. The Nth term that is calculated by the program should be written to STDOUT. Other than the value of Nth term, no other characters/strings or message should be written to STDOUT.
For example, when N = 14, the 14th term in the series is 17. So only the value 17 should be printed to STDOUT.

Solution in C

```
#include<stdio.h>
#define MAX 1000
void fibonacci(int n)
{
int i, t1 = 0, t2 = 1, nextTerm;
for (i = 1; i<=n; i++)
{
nextTerm = t1 + t2;
t1 = t2;
t2 = nextTerm;
}
printf("%d", t1);
}
void prime(int n)
{
int i, j, flag, count =0;
for (i=2; i<=MAX; i++)
{
flag = 0;
for (j=2; j<i; j++)
{
if(i%j == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
if(++count == n)
{
printf("%d", i);
break;
}
}
}
int main()
{
int n;
scanf("%d", &n);
if(n%2 == 1)
fibonacci (n/2 + 1);
else
prime(n/2);
```

```
return 0;
}
```

Solution in C++

```
#include<iostream>
using namespace std;
#define MAX 1000

void fibonacci(int n)
{
int i, t1 = 0, t2 = 1, nextTerm;
for (i = 1; i<=n; i++)
{
nextTerm = t1 + t2;
t1 = t2;
t2 = nextTerm;
}
cout << t1;
}
void prime(int n)
{
int i, j, flag, count =0;
for (i=2; i<=MAX; i++)
{
flag = 0;
for (j=2; j<i; j++)
{
if(i%j == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
if(++count == n)
{
cout << i;
break;
}
}
}
int main()
{
int n;
cin >> n;
if(n%2 == 1)
fibonacci (n/2 + 1);
else
prime(n/2);
return 0;
}
```

Problem 44:

Given a series whose even term creates a separate geometric series and odd term creates another geometric series. Write a program to generate such series.
For example,
1, 1, 2, 2, 4, 4, 8, 8, 16, 16,……

Solution in C

```
#include

int main()
{
int n, i, r1, r2;
printf("\nEnter the total number of terms : ");
scanf("%d", &n);
printf("\nEnter the common ratio for GP - 1 : ");
scanf("%d", &r1);
printf("\nEnter the common ratio for GP - 2 : ");
scanf("%d", &r2);
printf("\nThe series is\n");
int a = 1, b = 1;
if(n % 2 == 0)
{
for(i = 0; i < n/2; i++)
{
printf("%d ", a);
a = a * r1;
printf("%d ", b);
b = b * r2;
}
}
else
{
for(i = 0; i < n/2; i++)
{
printf("%d ", a);
a = a * r1;
printf("%d ", b);
b = b * r2;
}
printf("%d ", a);
}
printf("\n");

}
```

Solution in C++

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

int main()
{
int n, i, r1, r2;
cout << "\nEnter the total number of terms : ";
cin >> n;
```

```
cout << "\nEnter the common ratio for GP - 1 : ";
cin >> r1;
cout << "\nEnter the common ratio for GP - 2 : ";
cin >> r2;
cout << "\nThe series is\n";
int a = 1, b = 1;
if(n % 2 == 0)
{
for(i = 0; i < n/2; i++)
{
cout << a << " ";
a = a * r1;
cout << b << " ";
b = b * r2;
}
}
else
{
for(i = 0; i < n/2; i++)
{
cout << a << " ";
a = a * r1;
cout << b << " ";
b = b * r2;;
}
cout << a << " ";
}
cout << endl;
}
```

Program 45:

Program to find the factorial of a number using functions

C

```
#include <stdio.h>
int factorial_of_a_number(int n)
{
if(n == 0)
return 1;
if(n < 0)
printf("Invalid input\n");
int fact = 1, i;
for(i = 1; i <= n; i++)
{
fact = fact * i;
}
return fact;
}

int main()
{
int n;
printf("\nEnter the number : ");
scanf("%d",&n);
printf("\nFactorial of the number %d is %d" ,n, factorial_of_a_number(n));
printf("\n");
return 0;
}
```

C++

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

int factorial_of_a_number(int n)
{
if(n == 0)
return 1;
if(n < 0)
cout << "Invalid input\n";0
int fact = 1, i;
for(i = 1; i <= n; i++)
{
fact = fact * i;
}
return fact;
}

int main()
{
int n;
cout << "\nEnter the number : ";
cin >> n;
cout << "\nFactorial of the number " << n << " is " << factorial_of_a_number(n);
cout << endl;
```

```
return 0;
}
```

JAVA

```
import java.util.*;
public class Main
{
static int factorial_of_the_number(int n)
{
if(n < 0)
return -1;
if(n == 0)
return 1;
int fact = 1, i;
for(i = 1; i <= n; i++)
{
fact = fact * i;
}
return fact;
}
public static void main(String[] args)
{


int n;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter the number : ");
n = sc.nextInt();
System.out.println("Factorial of the number " + n + " is "+factorial_of_the_number(n));
}
}
```

Python

```
def factorial_of_the_number(n):
    if(n < 0):
        return -1
    if(n == 0):
        return 1
    fact = 1
    for i in range(1,n+1):
        fact = fact * i
    return fact


n = int(input("Enter the number : "))
print("Factorial of the number : ",end="")
print(factorial_of_the_number(n))
```

Program 46:

Program to find the factorial of a number using recursion

C

```
// Factorial of a number in C using recursion

#include <stdio.h>

int factorial_of_a_number(int n)
{
if(n < 0)
return -1;
if(n == 0)
return 1;
else
return (n * factorial_of_a_number(n-1));
}

int main()
{
int n;
printf("\nEnter the number : ");
scanf("%d",&n);
printf("\nFactorial of the number %d is %d\n",n,factorial_of_a_number(n));
return 0;
}
```

C++

```
// Factorial of a number in C++ using recursion

#include <iostream>
using namespace std;

int factorial_of_a_number(int n)
{
if(n < 0)
return -1;
if(n == 0)
return 1;
else
return (n * factorial_of_a_number(n-1));
}

int main()
{
int n;
cout << "\nEnter the number : ";
cin >> n;
cout << "\nFactorial of the number " << n << " is " << factorial_of_a_number(n);
cout << endl;
return 0;
}
```

JAVA

```
// Factorial of a number in java using recursion


import java.util.*;
public class Main
{
static int factorial_of_the_number(int n)
{
if(n < 0)
return -1;
if(n == 0)
return 1;
else
return (n * factorial_of_the_number(n – 1));
}
public static void main(String[] args)
{


int n;
Scanner sc = new Scanner(System.in);
System.out.print(“\nEnter the number : “);
n = sc.nextInt();
System.out.println(“Factorial of the number ” + n + ” is “+factorial_of_the_number(n));
}
}
```

Python

```
def factorial_of_the_number(n):
    if(n < 0):
        return -1
    if(n == 0):
        return 1
    else:
        return (n * factorial_of_the_number(n – 1))


n = int(input(“Enter the number : “))
print(“Factorial of the number : “,end=””)
print(factorial_of_the_number(n))
```

Program 47:

Program to find the factorial of a number using for loop and command line aruguments

```
#include
int main(int a, char *b[])  //command line arguments
{
int x,y,f=1;
x=atoi(b[1]);   //atoi function is to convert a character to integer
for(i=1;i<=x;i++)
{
f=f*i;
}
printf("%d",f);
return 0;
}
```

Program 48:

Program to find area of a circle using functions

C

```
// Area of a circle in C programming


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


float area_of_a_circle(float radius)
{
return M_PI * radius * radius;
}
int main()
{
float area, radius;
printf("\nEnter the radius of the circle : ");
scanf("%f", &radius);
area = area_of_a_circle(radius);
printf("\nArea of the circle : %f\n", area);
return 0;
```

C++

```
// C++ program to find the area of a circle


#include <iostream>
#include <math.h>
using namespace std;


float area_of_a_circle(float radius)
{
return M_PI * radius * radius;
}


int main()
{
float area, radius;
cout << "\nEnter the radius of the circle : ";
cin >> radius;
area = area_of_a_circle(radius);
cout << "\nArea of the circle : " << area <<endl;
return 0;
}
```

JAVA

```
// Java program to find the area of a circle
import java.util.*;
public class Main
{
static double area_of_a_circle(double radius)
{
return Math.PI * radius * radius;
```

```
}

public static void main (String[] args)
{
double radius;
Scanner sc = new Scanner(System.in);
System.out.print(“\nEnter the radius of the circle : “);
radius = sc.nextDouble();
System.out.println(“Area of the circle : ” + area_of_a_circle(radius));


}
}
```

Python

```
# Python program to find the area of a circle


from math import pi
def area_of_a_circle(radius):
return str(pi * radius**2)
radius = float(input (“Input the radius of the circle : “))
print (“The area of the circle is: ” + area_of_a_circle(radius))
```

Program 49:

Program to find area of a circle using pointers

C

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


float area_of_circle(float *radius)
{
return M_PI * (*radius) * (*radius);
}


#include<stdio.h>


int main()
{
float radius;


printf("\nEnter the radius of Circle : ");
scanf("%f", &radius);
printf("\nArea of Circle : %f", area_of_circle(&radius));


return (0);
}
```

C++

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


double area_of_circle(double *radius)
{
return M_PI * (*radius) * (*radius);
}


int main()
{
double radius;


cout << "\nEnter the radius of Circle : ";
cin >> radius;
cout << "\nArea of Circle : " << area_of_circle(&radius);
return 0;
}
```

Program 50:

Program to find area of circle using command line arguments

/*Here diameter is taken as the input*/

```
#include<stdio.h>
#define PI 3.14
int main(int a, char *b[])   //command line arguments
{
int d; float area =0;
d= atoi(argv[1]);
area =(float) PI*(d/2)*(d/2);
printf("%0.2f", area);    //%0.2f is to print the answer with 2 values after decimal point.
return 0;
}
```

Program 51:

Write a c program, to check whether the given year is a leap year or not. A leap year is a calendar year containing one additional day (Feb 29th) added to keep the calendar year synchronized with the astronomical year.

C

```
#include
int main(int a, char*b[])
{
int year; year=atoi(b[1]);
if(year%100==0)
{
if(year%400==0)
{
printf("LEAP YEAR");
}
else{
printf("NOT LEAP YEAR"); } }
else if(year%4==0)
{
printf("LEAP YEAR");
}
else{
printf("NOT LEAP YEAR");
}
return 0; }
```

OR

```
#include <stdio.h>


int main()
{
int year;
printf("Enter a year: ");
scanf("%d",&year);
printf("\n");
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
printf("%d is a leap year", year);
else
printf("%d is not a leap year", year);
}
else
printf("%d is a leap year", year );
}
else
printf("%d is not a leap year", year);
printf("\n");
return 0;
}
```

C++

```
#include <iostream>
using namespace std;
int main()
{
int year;
cout << “Enter a year: “;
cin >> year;
cout << endl;
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
cout << year << ” is a leap year” ;
else
cout << year << ” is not a leap year”;
}
else
cout << year << ” is a leap year”;
}
else
cout << year << ” is not a leap year”;
cout << endl;
return 0;
}
```

JAVA

```
import java.util.*;
public class Main
{
public static void main(String[] args) {
int year;
System.out.print(“Enter a year: “);
Scanner sc = new Scanner(System.in);
year = sc.nextInt();
System.out.println(“\n”);
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
System.out.println(year + ” is a leap year\n”);
else
System.out.println(year + ” is not a leap year\n”);
}
else
System.out.println(year + ” is a leap year\n”);
}
else
System.out.println(year + ” is not a leap year\n”);
}

}
```

Python

```
year = int(input(“Enter a year : “))
if(year%4 == 0):
    if( year%100 == 0):
        if ( year%400 == 0):
            print(year, end = ”)
            print(” is a leap year”)
        else:
            print(year, end = ”)
            print(” is not a leap year”)
    else:
        print(year, end = ”)
        print(” is a leap year”)
else:
    print(year, end = ”)
    print(” is not a leap year”)
```

Program 52:

Leap year program in C using functions

C

```
#include <stdio.h>

void is_leap_year(int year)
{
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
printf("%d is a leap year", year);
else
printf("%d is not a leap year", year);
}
else
printf("%d is a leap year", year);
}
else
printf("%d is not a leap year", year);
printf("\n");
}

int main()
{
int year;
printf("\nEnter a year : ");
scanf("%d",&year);
printf("\n");
is_leap_year(year);
return 0;
}
```

C++

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

void is_leap_year(int year)
{
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
cout << year << " is a leap year" ;
else
cout << year << " is not a leap year";
}
else
cout << year << " is a leap year";
}
else
```

```
cout << year << " is not a leap year";
cout << endl;
}

int main()
{
int year;
cout << "Enter a year: ";
cin >> year;
cout << endl;
is_leap_year(year);
return 0;
}
```

JAVA

```
import java.util.*;
public class Main
{
static void is_leap_year(int year)
{
if(year%4 == 0)
{
if( year%100 == 0)
{
if ( year%400 == 0)
System.out.println(year + " is a leap year\n");
else
System.out.println(year + " is not a leap year\n");
}
else
System.out.println(year + " is a leap year\n");
}
else
System.out.println(year + " is not a leap year\n");
}

public static void main(String[] args) {
int year;
System.out.print("Enter a year: ");
Scanner sc = new Scanner(System.in);
year = sc.nextInt();
System.out.println("\n");
is_leap_year(year);
}

}
```

Python

```
def is_leap_year(year):
    if(year%4 == 0):
        if( year%100 == 0):
            if ( year%400 == 0):
                print(year, end = '')
                print(" is a leap year")
            else:
                print(year, end = '')
```

```
            print(” is not a leap year”)
        else:
            print(year, end = ”)
            print(” is a leap year”)
    else:
        print(year, end = ”)
        print(” is not a leap year”)


year = int(input(“Enter a year : “))
is_leap_year(year)
```

Program 53:

Program to find the GCD or HCF of two numbers

```
#include
int main(int x, char *y[])
{
inta,b,small,i;
a=atoi(y[1]);
b=atoi(y[2]);
small=a>b?b:a;
for(i=small;i>=1;i–)
{
if((a%i==0)&&(b%i==0))
{
printf(“%d”,i);
break;
} }
return 0;
}
```

Or

C

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int a,b,gcd;
printf(“\nEnter two numbers : “);
scanf(“%d %d”,&a,&b);
int i;
for(i = 1; i <= a && i <= b; i++)
{
if((a % i == 0) && (b % i == 0))
{
gcd = i;
}
}
printf(“\nGCD of %d and %d is %d “,a,b,gcd);
printf(“\n”);
return 0;
}
```

C++

```
#include<iostream>
using namespace std;
int main()
{
int a,b,gcd;
cout <<“\nEnter two numbers : “;
cin >> a >> b;
int i;
for(i = 1; i <= a && i <= b; i++)
{
if((a % i == 0) && (b % i == 0))
{
gcd = i;
```

```
}
}
cout << "\nGCD of "<< a << " and " << b << " is " << gcd;
cout << endl;
return 0;
}
```

JAVA

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int a,b,gcd = 0;
System.out.print("\nEnter two numbers : ");
Scanner sc = new Scanner(System.in);
a = sc.nextInt();
b = sc.nextInt();
int i;
for(i = 1; i <= a && i <= b; i++)
{
if((a % i == 0) && (b % i == 0))
{
gcd = i;
}
}
System.out.println("\nGCD of " + a + " and " + b + " is " + gcd);
System.out.println();
}
}
```

Python

```
gcd = 0
print("Enter two numbers : ")
a = int(input())
b = int(input())
i = 1
    while(i <= a and i <= b):
        if((a % i == 0) and (b % i == 0)):
            gcd = i
        i = i + 1


print("GCD : ",end="");
print(gcd)
```

Program 54:

GCD of two numbers using recursion

C

```
#include <stdio.h>


int gcd(int a, int b)
{
if (b != 0)
return gcd(b, a % b);
else
return a;
}


int main()
{
int a, b;
printf("Enter two numbers: ");
scanf("%d %d", &a, &b);


printf("\nGCD of %d and %d is %d\n", a, b, gcd(a,b));
return 0;
}
```

C++

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


int gcd(int a, int b)
{
if (b != 0)
return gcd(b, a % b);
else
return a;
}


int main()
{
int a, b;
cout << "Enter two numbers: ";
cin >> a >> b;
cout << "\nGCD of " << a << " and " << b << " is " << gcd(a,b);
cout << endl;
return 0;
}
```

JAVA

```
import java.util.*;
public class Main
{
static int gcd(int a, int b)
{
```

```
if (b != 0)
return gcd(b, a % b);
else
return a;
}

public static void main(String[] args)
{
int a,b,gcd = 0;
System.out.print(“\nEnter two numbers : “);
Scanner sc = new Scanner(System.in);
a = sc.nextInt();
b = sc.nextInt();
System.out.println(“\nGCD of ” + a + ” and ” + b + ” is ” + gcd(a,b));
System.out.println();
}}
```

Python

```
def gcd(a, b):
    if (b != 0):
        return gcd(b, a % b)
    else:
        return a

print(“Enter two numbers : “)
a = int(input())
b = int(input())

gcd = gcd(a,b)
print(“GCD : “,end = “”);
print(gcd)
```

Program 55:

Check whether a given number is a prime or not

C (using for loop)

```
// C program to check if the given  number is prime or not

#include<stdio.h>

int main()
{
int n,i;
printf("\nEnter the number : ");
scanf("%d",&n);

for(i = 2; i <= n/2; i++)
{
if(n % i ==0)
{
break;
}
}
if(i > n/2)
printf("\n%d is a Prime Number\n",n);
else
printf("\n%d is not a Prime Number\n", n);
return 0;
}
```

C (using pointers)

```
// Prime number program in c using pointers

#include<stdio.h>

int is_prime_number(int *p)
{
int i;
for(i = 2; i <= *p/2; i++)
{
if(*p % i ==0)
{
break;
}
}
if(i > *p/2)
printf("\n%d is a Prime Number\n",*p);
else
printf("\n%d is not a Prime Number\n", *p);
}
int main()
{
int n,i;
printf("\nEnter the number : ");
scanf("%d",&n);
```

```
int *p = &n;
is_prime_number(p);
return 0;
}
```

C (using functions)

```
// C  program to check if the given  number is prime or not


#include<stdio.h>


int is_prime_number(int n)
{
int i;
for(i = 2; i <= n/2; i++)
{
if(n % i ==0)
{
break;
}
}
if(i > n/2)
printf("\n%d is a Prime Number\n",n);
else
printf("\n%d is not a Prime Number\n", n);
}
int main()
{
int n,i;
printf("\nEnter the number : ");
scanf("%d",&n);
is_prime_number(n);
return 0;
}
```

C (using recursion)

```
#include<stdio.h>
int is_prime_number(int num, int i)
{
if(num < 2)
{
printf("\nEnter numbers greater than 1\n");
exit(0);
}
if (i == 1)
{
return 1;
}
else
{
if (num % i == 0)
{
return 0;
}
else
{
```

```
return is_prime_number(num, i – 1);
}
}
}

int main()
{
int n, flag;
printf(“Enter a number: “);
scanf(“%d”, &n);

flag = is_prime_number(n, n / 2);
if (flag == 1)
{
printf(“\n%d is a prime number\n”, n);
}
else
{
printf(“\n%d is not a prime number\n”, n);
}
return 0;
}
```

C (using command line)

```
#include<stdio.h>
#include int main(int a, char *b[])
{
int number,i,flag = 1;
number = atoi(b[1]);
for(i=2; i<number; i++)
{
if(number%i == 0)
{
flag = 0;
break;
}
}
if(flag == 1)
printf(“%.2f”,sqrt(number));
else
printf(“0.00”);
return 0;
}
```

Program 56:

Program to find prime numbers in a given range using loops

C

// C program to find prime numbers in a given range

```
#include <stdio.h>
int main()
{
int a, b, i, flag;
printf("\nEnter start value : ");
scanf("%d",&a);
printf("\nEnter end value : ");
scanf("%d",&b);
printf("\nPrime Numbers between %d and %d : ", a, b);
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
printf("%d ", a);
++a;
}
printf("\n");
return 0;
}
```

C++

C++ program to find prime numbers in a given range

```
#include <iostream>
using namespace std;
int main()
{
int a, b, i, flag;
cout << "\nEnter start value : ";
cin >> a;
cout << "\nEnter end value : " ;
cin >> b;
cout << "\nPrime Numbers between " << a << " and " << b <<" : ";
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
```

```
flag = 1;
break;
}
}
if (flag == 0)
cout << a << " ";
++a;
}
cout << endl;
return 0;
}
```

JAVA

```
// Java program to find prime numbers in a given range


import java.util.*;
public class Main
{
public static void main(String args[])
{
int a,b,flag,i;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter start value : ");
a = sc.nextInt();
System.out.print("\nEnter end value : ");
b = sc.nextInt();
System.out.print("\nPrime numbers between "+ a + " and " + b + " are : ");
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
System.out.print(a + " ");
++a;
}
}
}
```

Python

```
# Python program to find prime numbers in a given range


a = int(input("Enter start value : "))
b = int(input("Enter end value : "))
print("Prime numbers between",a,"and",b,"are :")
while (a < b):
flag = 0;
for i in range(2, int(a/2),1):
if(a % i == 0):
```

```
flag = 1;
break
if (flag == 0):
print(a, end = ” “)
a = a + 1
```

Command Line

```
#include<stdio.h>
int main(int argc, char *argv[])
{
int N1, N2, j, i, count, sum = 0;
N1 =atoi(argv[1]);
N2 =atoi(argv[2]);
for(i=N1+1; i<N2; ++i)
{
count = 0;
for(j=2; j<=(i/2); j++)
{
if(i%j==0)
{
count++;
break;
}
}
if(count==0)
sum = sum + i;
}
printf(“%d”,sum);
return 0;
}
```

Program 57:

Program to find prime numbers in a given range using functions

C

```
// Prime numbers from 1 to n in C


#include <stdio.h>


bool is_prime_number(int n)
{
if (n <= 1)
return false;
if (n <= 3)
return true;


if (n % 2 == 0 || n % 3 == 0)
return false;


for (int i = 5; i * i <= n; i = i + 6)
if (n % i == 0 || n % (i + 2) == 0)
return false;


return true;
}


void print_prime_numbers(int n)
{
for (int i = 2; i <= n; i++)
{
if (is_prime_number(i))
printf("%d ",i);
}
}
int main()
{
int n;
printf("\nEnter the end value : ") ;
scanf("%d",&n);
printf("\nThe Prime Numbers are : ");
print_prime_numbers(n);
printf("\n");
}
```

C++

```
// Prime numbers from 1 to n in C++


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


bool is_prime_number(int n)
{
```

```
if (n <= 1)
return false;
if (n <= 3)
return true;

if (n % 2 == 0 || n % 3 == 0)
return false;

for (int i = 5; i * i <= n; i = i + 6)
if (n % i == 0 || n % (i + 2) == 0)
return false;

return true;
}

void print_prime_numbers(int n)
{
for (int i = 2; i <= n; i++) {
if (is_prime_number(i))
cout << i << " ";
}
}
int main()
{
int n;
cout << "\nEnter the end value : ";
cin >> n;
cout << "\nThe Prime Numbers are : ";
print_prime_numbers(n);
cout << endl;
}
```

JAVA

```
// Prime numbers from 1 to n in java

import java.util.*;

public class Main
{
static boolean is_prime_number(int n)
{
if (n <= 1)
return false;
if (n <= 3)
return true;

if (n % 2 == 0 || n % 3 == 0)
return false;

for (int i = 5; i * i <= n; i = i + 6)
if (n % i == 0 || n % (i + 2) == 0)
return false;
```

```
return true;
}

static void print_prime_numbers(int n)
{
for (int i = 2; i <= n; i++)
{
if (is_prime_number(i))
System.out.print(i + " ");
}
}

public static void main(String args[])
{
int n;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter end value : ");
n = sc.nextInt();
System.out.print("\nThe Prime numbers are ");
print_prime_numbers(n);
}
}
```

Python

```
# Prime numbers from 1 to n in python

def is_prime_number(n):
if (n <= 1):
return False
if (n <= 3):
return True


if (n % 2 == 0 or n % 3 == 0):
return False;


i = 5
while(i * i <= n) :
if (n % i == 0 or n % (i + 2) == 0) :
return False
i = i + 6

return True

def print_prime_numbers(n):
for i in range (2,n+1,1):
if (is_prime_number(i)):
print(i, end = " ");


n = int(input("Enter end value : "))
print("The Prime numbers are :",end = " ")
print_prime_numbers(n)
```

Program 58:

Program to check if a given number is a strong number or not

C

```
// C program to check if a given number is a strong number or not

#include<stdio.h>

int main()
{
int n,i;
int fact,rem;
printf("\nEnter a number : ");
scanf("%d",&n);
printf("\n");
int sum = 0;
int temp = n;
while(n)
{
i = 1,fact = 1;
rem = n % 10;

while(i <= rem)
{
fact = fact * i;
i++;
}
sum = sum + fact;
n = n / 10;
}
if(sum == temp)
printf("%d is a strong number\n",temp);
else
printf("%d is not a strong number\n",temp);

return 0;
}
```

C++

```
// C++ program to check if a given number is a strong number or not

#include<iostream>
using namespace std;

int main()
{
int n,i;
int fact,rem;
cout << "\nEnter a number : ";
cin >> n;
cout << endl;
int sum = 0;
```

```
int temp = n;
while(n)
{
i = 1,fact = 1;
rem = n % 10;


while(i <= rem)
{
fact = fact * i;
i++;
}
sum = sum + fact;
n = n / 10;
}
if(sum == temp)
    cout << temp << " is a strong number\n";
else
    cout << temp << " is not a strong number\n";


return 0;
}
```

JAVA

```
// Java program to check if a given number is a strong number or not


import java.util.*;
public class Main
{
public static void main(String[] args) {
int n,i;
int fact,rem;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter the number : ");
n = sc.nextInt();
int sum = 0;
int temp = n;
while(n != 0)
{
i = 1;
fact = 1;
rem = n % 10;


while(i <= rem)
{
fact = fact * i;
i++;
}
sum = sum + fact;
n = n / 10;
}


if(sum == temp)
System.out.println(temp + " is a strong number\n");
```

```
else
System.out.println(temp + ” is not a strong number\n”);


System.out.println();
}
}
```

Python

```
// Python program to check if a given number is a strong number or not


n = int(input(“Enter the number : “))
sum = 0
temp = n
while (n):
    i = 1
    fact = 1
    rem = int(n % 10)


while(i <= rem):
    fact = fact * i
    i = i + 1
    sum = sum + fact
    n = int(n / 10)


if(sum == temp):
    print(temp,end = “”)
    print(” is a strong number”)
else:
    print(temp,end = “”)
    print(” is not a strong number”)
```

Program 59:

Strong number program using command line arguments

```
#include<stdio.h>
#include<stdlib.h>
int main(int a, char *b[])
{
int number, i, temp, sum = 0, factorial = 1;
number = atoi(b[1]);
temp = number;
while(number != 0)
{
int rem = number%10;
for(i=2; i<=rem; i++)
{
factorial = factorial * i;
}
sum = sum + factorial;
number = number/10;
factorial = 1;
}
if(temp == sum)
printf("YES");
else
printf("NO");
return 0;
}
```

Program 60:

Check whether a number is PALINDROME or Not (Palindrome or not using iterative approach)

C

```
// C program to check if the given number is a palindrome or not

#include <stdio.h>
int main()
{
int n, reverse = 0, remainder, number;

printf("Enter an integer: ");
scanf("%d", &n);

number = n;

while( n!=0 )
{
remainder = n%10;
reverse = reverse*10 + remainder;
n /= 10;
}

if (number == reverse)
printf("\n%d is a palindrome\n", number);
else
printf("\n%d is not a palindrome\n", number);

return 0;
}
```

C++

```
// C++ program to check if the given number is a palindrome or not

#include <iostream>
using namespace std;
int main()
{
int n, reverse = 0, remainder, number;

cout << "Enter an integer: ";
cin >> n;

number = n;

while( n!=0 )
{
remainder = n%10;
reverse = reverse*10 + remainder;
```

```
n /= 10;
}

if (number == reverse)
cout << number << " is a palindrome\n";
else
cout << number << " is not a palindrome\n";

return 0;
}
```

JAVA

```
// Java program to check if the given number is a palindrome or not

import java.util.*;
public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
int n, reverse = 0, remainder, number;
System.out.print("Enter a number : ");
n = sc.nextInt();
number = n;
while( n!=0 )
{
remainder = n%10;
reverse = reverse*10 + remainder;
n /= 10;
}
if(number == reverse)
{
System.out.println(number + " is a palindrome\n");
}
else
{
System.out.println(number + " is not a palindrome\n");
}

}
}
```

Python

```
# Python program to check if the given number is a palindrome or not

n = int(input("Enter the binary number : "))
reverse = 0
number = n
while( n!=0 ):
remainder = n%10
reverse = reverse*10 + remainder
n =int(n/10)
if(number == reverse):
```

```
print(number, “is a palindrome\n”)
else:
print(number, “is not a palindrome\n”)
```

Command Line

```
#include<stdio.h>
#include<math.h>
int main(int a,int *b[])
{
int  number, rem, sum = 0;
number = atoi(b[1]);
int copy = number;
while(number != 0)
{
rem =number%10;
sum = sum * 10 + rem;
number = number/10;
}
if(copy == sum)
printf(“Palindrome”);
else
printf(“Not Palindrome”);
return 0;
}
```

Program 61:

Palindrome or not using recursive approach

C

```
#include<stdio.h>

int is_Palindrome(int );

int n;

int main()
{
int palindrome;
printf("\n\nEnter a number : ");
scanf("%d", &n);
palindrome = is_Palindrome(n);
if(palindrome == 1)
printf("\n%d is palindrome\n", n);
else
printf("\n%d is not palindrome\n", n);

return 0;
}

int is_Palindrome(int aj)
{
static int sum = 0;
if(aj != 0)
{
sum = sum *10 + aj%10;
is_Palindrome(aj/10); // recursive call
}
else if(sum == n)
return 1;
else
return 0;
}
```

C++

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

int is_Palindrome(int );

int n;

int main()
{
int palindrome;
cout << "\n\nEnter a number : ";
cin >> n;
```

```
cout << endl;
palindrome = is_Palindrome(n);
if(palindrome == 1)
cout << n << " is a palindrome\n";
else
cout << n << " is not a palindrome\n";
return 0;
}

int is_Palindrome(int aj)
{
static int sum = 0;
if(aj != 0)
{
sum = sum *10 + aj%10;
is_Palindrome(aj/10); // recursive call
}
else if(sum == n)
return 1;
else
return 0;
}
```

JAVA

```
import java.util.*;

class Main
{
static int rev(int n, int temp)
{
// base case
if (n == 0)
return temp;

// stores the reverse of the number
temp = (temp * 10) + (n % 10);

return rev(n / 10, temp);
}

public static void main (String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter a number : ");
n = sc.nextInt();
int temp = rev(n, 0);

if (temp == n)
System.out.println("yes");
else
System.out.println("no" );
}
}
```

Python

```
def rev(n, temp):
# base case
if (n == 0):
return temp;
# stores the reverse of a number
temp = (temp * 10) + (n % 10);
return rev(n / 10, temp);

n = 454;
temp = rev(n, 0);
if (temp != n):
print(“yes”);
else:
print(“no”);
```

Problem 62:

Program to check whether the given string is a palindrome or not

C

```
/* C program to check if the given string is a palindrome or not */
#include <stdio.h>
#include <string.h>


int main()
{
char a[100], b[100];


printf("Enter the string : ");
gets(a);


strcpy(b, a); /* Copying input string */
strrev(b); /* Reversing the string */


if (strcmp(a, b) == 0) /* Comparing input string with the reverse string */
printf("The string is a palindrome\n");
else
printf("The string is not t a palindrome\n");


return 0;
}
```

C++

```
/* C++ program to check if the given string is a palindrome or not */
#include <iostream>
#include <string.h>
using namespace std;


int main()
{
char a[100], b[100];


cout << "Enter the string : ";
cin >> a;


/* Reversing the string */
int i, n = strlen(a);
for(i = 0; i < n; i++)
{
b[n-1-i] = a[i];
}


if (strcmp(a, b) == 0) /* Comparing input string with the reverse string */
cout << "The string is a palindrome\n";
else
cout << "The string is not a palindrome\n";
```

```
return 0;
}
```

JAVA

```
/* Java program to check whether a string is a palindrome or not */


import java.util.*;
public class Main
{
public static void main(String args[])
{
String a, b = “”;
Scanner s = new Scanner(System.in);
System.out.print(“Enter the string : “);
a = s.nextLine();
int n = a.length();
for(int i = n – 1; i >= 0; i–)
{
b = b + a.charAt(i);
}
if(a.equalsIgnoreCase(b))
{
System.out.println(“The string is a palindrome”);
}
else
{
System.out.println(“The string is not a palindrome”);
}
}
}
```

Python

```
# Python program to check whether a string is a palindrome or not


string = input(“Enter the string:”)
if(string == string[::-1]):
print(“The string is a palindrome”)
else:
print(“The string is not a palindrome”)
```

Problem 63:

Program to check whether the number is Armstrong or not

C

```
// C program to check whether the given number is Armstrong or not


#include
#include


int main()
{
int number, temp, remainder, result = 0, n = 0 ;


printf(“Enter an integer: “);
scanf(“%d”, &number);


temp = number;


// Finding the number of digits


while (temp != 0)
{
temp /= 10;
++n;
}


temp = number;


// Checking if the number is armstrong


while (temp != 0)
{
remainder = temp%10;
result += pow(remainder, n);
temp /= 10;
}


if(result == number)
printf(“%d is an Armstrong number\n”, number);
else
printf(“%d is not an Armstrong number\n”, number);


return 0;
}
```

C++

```
// C++ program to check whether the number is Armstrong or not
```

```
#include
#include using namespace std;

int main()
{
int number, temp, remainder, result = 0, n = 0 ;

cout << “Enter an integer : “;
cin >> number;

temp = number;

// Finding the number of digits

while (temp != 0)
{
temp /= 10;
++n;
}

temp = number;

// Checking if the number is armstrong

while (temp != 0)
{
remainder = temp%10;
result += pow(remainder, n);
temp /= 10;
}

if(result == number)
cout << number << ” is an Armstrong number\n”;
else
cout << number << ” is not an Armstrong number\n”;

return 0;
}
```

JAVA

```
// Java program to check whether the number is Armstrong or not

import java.util.*;
public class sum_of_primes {

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print(“Enter a number: “);
int number = sc.nextInt();
```

```
int temp, remainder, result = 0, n = 0 ;
temp = number;

// Finding the number of digits

while (temp != 0)
{
temp /= 10;
++n;
}

temp = number;

// Checking if the number is armstrong

while (temp != 0)
{
remainder = temp%10;
result += Math.pow(remainder, n);
temp /= 10;
}

if(result == number)
System.out.print(number + ” is an Armstrong number\n”);
else
System.out.print(number + ” is not an Armstrong number\n”);

}

}
```

Python

```
# Python program to check whether the number is Armstrong or not

import math
number = int(input(“Enter a number : “))
result = 0
n = 0
temp = number;
while (temp != 0):
temp =int(temp/10)
n = n + 1

// Checking if the number is armstrong

temp = number
while (temp != 0):
remainder = temp % 10
result = result + math.pow(remainder, n)
temp =int(temp/10)
```

```
if(result == number):
print(number,”is an Armstrong number”)
else:
print(number,”is not an Armstrong number”)
```

Command Line

```
#include<stdio.h>
#include<math.h>
#include int main(int a, char*b[])
{
int n;
n= atoi(b[1]);
int sum=0;
int temp=n;
int cnt=0;
while(n!=0)
{
n=n/10;
cnt++;
}
n=temp;
while(n!=0)
{
int rem=n%10;
sum=sum+pow(rem,cnt);
n=n/10;
}
if(temp==sum)
{
printf(“yes”);
}
else
{
printf(“no”);
}
return 0;
}
```

Problem 64:

Program to print the Armstrong numbers between the two intervals

C

```
// C program to print the Armstrong numbers between the two intervals


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


int main()
{
int start, end, i, temp1, temp2, remainder, n = 0, result = 0;


printf("Enter start value and end value : ");
scanf("%d %d", &start, &end);
printf("\nArmstrong numbers between %d an %d are: ", start, end);


for(i = start + 1; i < end; ++i)
{
temp2 = i;
temp1 = i;


while (temp1 != 0)
{
temp1 /= 10;
++n;
}


while (temp2 != 0)
{
remainder = temp2 % 10;
result += pow(remainder, n);
temp2 /= 10;
}


if (result == i) {
printf("%d ", i);
}


n = 0;
result = 0;


}
printf("\n");
return 0;
}
```

C++

```
// C++ program to print the Armstrong numbers between the two intervals
```

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

int main()
{
int start, end, i, temp1, temp2, remainder, n = 0, result = 0;

cout << “Enter start value and end value : “;
cin >> start >> end;
cout << “\nArmstrong numbers between ” << start << ” and ” << end << ” are : “;

for(i = start + 1; i < end; ++i)
{
temp2 = i;
temp1 = i;

while (temp1 != 0)
{
temp1 /= 10;
++n;
}

while (temp2 != 0)
{
remainder = temp2 % 10;
result += pow(remainder, n);
temp2 /= 10;
}

if (result == i) {
cout << i << ” “;
}

n = 0;
result = 0;

}
cout << endl;
return 0;
}
```

JAVA

```
//Java program to print the Armstrong numbers between the two intervals

import java.util.*;
public class sum_of_primes {

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
```

```
System.out.print(“Enter start and end values: “);
int start = sc.nextInt();
int end = sc.nextInt();
int i, temp1, temp2, remainder, n = 0, result = 0;



for(i = start + 1; i < end; ++i)
{
temp2 = i;
temp1 = i;


while (temp1 != 0)
{
temp1 /= 10;
++n;
}


while (temp2 != 0)
{
remainder = temp2 % 10;
result += Math.pow(remainder, n);
temp2 /= 10;
}


if (result == i) {
System.out.print(i + ” “);
}


n = 0;
result = 0;


}


}


}
```

Python

```
# Python program to print the Armstrong numbers between the two intervals


import math
start = int(input(“Enter start value : “))
end = int(input(“Enter end value : “))
result = 0
n = 0
for i in range(start+1 ,end,1):
temp2 = i
temp1 = i


while (temp1 != 0):
temp1 = int(temp1/10)
```

```
n = n + 1

while (temp2 != 0):
remainder = temp2 % 10
result = result + math.pow(remainder, n)
temp2 = int(temp2/10)


if (result == i):
print(i," ")

n = 0
result = 0
```

Problem 65:

Program to generate Fibonacci series upto n value

C

```
// C program to generate fibonacci series upto n value


#include


int main()
{
int i,c=0,n;
int a=0;
int b=1;
printf("\nEnter the nth value : ");
scanf("%d",&n);
printf("\nFibonacci series : ");
while(c<=n)
{
printf("%d ",c);
a=b;   // swap elements
b=c;
c=a+b; // next term is the sum of the last two terms
}
printf("\n");
return 0;
}
```

C++

```
// C++ program to generate fibonacci series upto n value


#include<iostream>
using namespace std;


int main()
{
int i,c=0,n;
int a=0;
int b=1;
cout << "\nEnter the nth value : ";
cin >> n;
cout << "\nFibonacci series : ";
while(c<=n)
{
cout << c << " ";
a=b;
b=c;
c=a+b;
}
cout << endl;
return 0;
}
```

JAVA

```
// Java program to generate fibonacci series upto n value


import java.util.*;
public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the nth value : ");
int n = sc.nextInt();
int a=0;
int b=1;
int c = a+b;
System.out.print("\nFibonacci series : ");
while(c<=n)
{
System.out.print(c + " " );
a=b;
b=c;
c=a+b;
}
}
}
```

Python

```
# Python program to generate Fibonacci series upto n value
n = int(input("Enter the nth value: "))
a = 0
b = 1
c = 0
print("Fibonacci Series : ",end = " ")
while(c <=n):
print(c,end = " ")
a = b
b = c
c = a + b
```

Command Line

```
#include<stdio.h>
#include<math.h>
int main(int a, char *b[])
{
int i, n, t1 = 0, t2 = 1, nextTerm;
n=atoi(b[1]);
for (i = 1; i <= n; ++i)
{
printf("%d ", t1);
nextTerm = t1 + t2;
t1 = t2;
t2 = nextTerm;
}
return 0;
}
```

Problem 66:

Program to convert the given binary number into decimal number

C

```
// C program to convert a binary number into decimal number


#include
#include


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;
printf("Enter a binary number: ");
scanf("%ld", &n);
printf("\nDecimal number : %d\n ", binary_to_decimal(n));
return 0;
}
```

C++

```
// C++ program to convert a binary number into decimal number


#include
#include using namespace std;


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;
cout << “Enter a binary number: “;
cin >> n;
cout << “\nDecimal number : ” << binary_to_decimal(n) << endl;
return 0;
}
```

JAVA

```
// Java program to convert abinary number into decimal number


import java.util.*;
public class binary_to_decimal {


public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print(“Enter the binary number : “);
int n = sc.nextInt();
int decimal = 0, i = 0, remainder;
while (n!=0)
{
remainder = n%10;
n /= 10;
decimal += remainder*Math.pow(2,i);
++i;
}
System.out.println(“Decimal number : ” + decimal);
}
}
```

Python

```
# Python program to convert a binary number into decimal number


import math
n = int(input(“Enter the binary number : “))
decimal = 0
remainder = 0
i = 0
while (n!=0):
remainder = n%10
n = int(n / 10)
decimal = decimal + remainder*math.pow(2,i)
i = i + 1
print(“Decimal Number : “,int(decimal))
```

Problem 67:

Program to convert numbers from binary numbers of higher range to decimal

C++

```
// C++ program to convert binary to decimal
#include <iostream>
#include <string>
using namespace std;


// Function to convert binary to decimal
int binary_to_decimal(string n)
{
string num = n;
int decimal = 0;


// Initializing base value to 1 which is (2^0)
int base = 1;


int len = num.length();
for (int i = len – 1; i >= 0; i–) {
if (num[i] == ‘1’)
decimal+= base;
base = base * 2;
}


return decimal;
}


int main()
{
string num;
cout <<“\nEnter the binary number : “;
cin >> num;
cout << “\nDecimal Equivalent : ” << binary_to_decimal(num) << endl;
}
```

JAVA

```
// Java program to convert binary to decimal


import java.io.*;


class Main {


// Function to convert binary to decimal
static int binaryToDecimal(String n)
{
String num = n;
int dec_value = 0;
```

```
// Initializing base value to 1 which is (2^0)
int base = 1;

int len = num.length();
for (int i = len – 1; i >= 0; i–) {
if (num.charAt(i) == ‘1’)
dec_value += base;
base = base * 2;
}

return dec_value;
}

public static void main(String[] args)
{
String num = new String(“1111”);
System.out.println(binaryToDecimal(num));
}
}
```

Python

```
# Python program to convert binary to decimal

def binaryToDecimal(n):
num = n;
dec_value = 0;

# Initializing base
# value to 1, which is 2 ^ 0
base1 = 1;

len1 = len(num);
for i in range(len1 – 1, -1, -1):
if (num[i] == ‘1’):
dec_value += base1;
base1 = base1 * 2;

return dec_value;

num = “1111”;
print(binaryToDecimal(num));
```

Problem 68:

Program to convert a number from decimal to binary

C

```
// C program to convert a number from decimal to binary


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


long int decimal_to_binary(int n)
{
long int binary = 0;
int remainder, i = 1, flag = 1;


while (n!=0)
{
remainder = n%2;
n /= 2;
binary += remainder*i;
i *= 10;
}
return binary;
}


int main()
{
int n;
printf("\nEnter a decimal number: ");
scanf("%d", &n);
printf("\nEquivalent binary number : %d\n", decimal_to_binary(n));
return 0;
}
```

C++

```
// C++ program to convert a number from decimal to binary


#include <iostream>
#include <math.h>
using namespace std;


long int decimal_to_binary(int n)
{
long int binary = 0;
int remainder, i = 1, flag = 1;


while (n!=0)
{
remainder = n%2;
n /= 2;
binary += remainder*i;
i *= 10;
```

```
}
return binary;
}

int main()
{
int n;
cout << "\nEnter a decimal number: ";
cin >> n;
cout << "\nEquivalent binary number : " << decimal_to_binary(n) << endl;
return 0;
}
```

JAVA

```
// Java program to convert a number from decimal to binary

import java.util.*;
public class decimal_to_binary
{
public static int decimal_to_binary(int n)
{
int binary = 0;
int remainder, i = 1, flag = 1;

while (n!=0)
{
remainder = n%2;
n /= 2;
binary += remainder*i;
i *= 10;
}
return binary;
}

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the decimal number : ");
int n = sc.nextInt();
System.out.println("Equivalent binary number : " + decimal_to_binary(n));
}
}
```

Python

```
# Python program to convert a number from decimal to binary

import math
def decimal_to_binary(n):
binary = 0
i = 1
flag = 1
while (n!=0):
remainder = n%2
```

```
n = int(n/2)
binary = binary + remainder*i
i = i * 10
return binary

n = int(input(“Enter the binary number : “))
print(“Equivalent binary number : “,decimal_to_binary(n))
```

Command Line

```
#include<stdio.h>
#include<math.h>
int main(int a, char *argv[])
{
int number, count, i;
int b[32];
number = atoi(argv[1]);
count = 0;
while(number != 0)
{
b[count]=number%2;
number = number/2;
count++;
}
for(i=(count-1); i>=0; i–)
printf(“%d”, b[i]);
return 0;
}
```

Problem 69:

Program to convert a number from decimal to octal

C

```
// C program to convert a number from decimal to octal

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

int decimal_to_octal(int decimal);
int main()
{
int decimal;

printf("\nEnter a decimal number: ");
scanf("%d", &decimal);

printf("\nEquivalent octal number : %d\n", decimal_to_octal(decimal));

return 0;
}

int decimal_to_octal(int decimal)
{
int octal = 0, i = 1;

while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}

return octal;
}
```

C++

```
// C++ program to convert a number from decimal to octal

#include <iostream>
#include <math.h>
using namespace std;

int decimal_to_octal(int decimal)
{
int octal = 0, i = 1;
```

```
while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}

return octal;
}

int main()
{
int decimal;

cout << "\nEnter a decimal number: ";
cin >> decimal;

cout << "\nEquivalent octal number : " << decimal_to_octal(decimal);
cout << endl;

return 0;
}
```

JAVA

```
// Java program to convert a number from decimal to octal

import java.util.*;
public class Main
{
public static int decimal_to_octal(int decimal)
{
int octal = 0, i = 1;

while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}

return octal;
}

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the decimal number : ");
int decimal = sc.nextInt();
System.out.print("\nEquivalent octal number : " + decimal_to_octal(decimal));
}
}
```

Python

```
# Python program to convert a number from decimal to octal


def decimal_to_octal(decimal):
octal = 0
i = 1
while (decimal != 0):
octal = octal + (decimal % 8) * i
decimal = int(decimal / 8)
i = i * 10;


return octal;


decimal = int(input(“Enter the decimal number : “))
print(“Equivalent octal number : “,decimal_to_octal(decimal))
```

Problem 70:

Program to convert a number from octal to decimal

C

```
// C program to convert a number from octal to decimal

#include
#include

long int octal_to_decimal(int octal)
{
int decimal = 0, i = 0;

while(octal != 0)
{
decimal += (octal%10) * pow(8,i); // multiplying with powers of 8
++i;
octal/=10;  // Divide by 10 to make it as decimal
}

i = 1;

return decimal;
}

int main()
{
int octal;

printf("\nEnter an octal number: ");
scanf("%d", &octal);

printf("\nDecimal Equivalent : %d\n",octal_to_decimal(octal));

return 0;
}
```

C++

```
// C++ program to convert a number from octal to decimal

#include
#include using namespace std;

long int octal_to_decimal(int octal)
{
int decimal = 0, i = 0;
```

```
while(octal != 0)
{
decimal += (octal%10) * pow(8,i);  // multiply with powers of 8
++i;
octal/=10;
}

i = 1;

return decimal;
}

int main()
{
int octal;

cout << "\nEnter an octal number: ";
cin >> octal;

cout << "\nDecimal Equivalent : " << octal_to_decimal(octal) << endl;;

return 0;
}
```

JAVA

```
// Java program to convert a number from octal to decimal

import java.util.*;
public class Main
{
public static int octal_to_decimal(int octal)
{
int decimal = 0, i = 0;

while(octal != 0)
{
decimal += (octal%10) * Math.pow(8,i);
++i;
octal/=10;
}

i = 1;

return decimal;
}



public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the octal number : ");
```

```
int octal = sc.nextInt();
System.out.print("\nEquivalent decimal number : " + octal_to_decimal(octal));
}
}
```

Python

```
# Python program to convert a number from octal to decimal


import math
def octal_to_decimal(octal):
decimal = 0
i = 0
while(octal != 0):
decimal += (octal%10) * math.pow(8,i)
i = i + 1
octal = int(octal / 10)
i = 1;


return decimal


octal = int(input("Enter the octal number : "))
print("Equivalent decimal number : ",int(octal_to_decimal(octal)))
```

Problem 71:

Program to convert a number from binary to octal

C

```
// C program to convert a number from binary to octal

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

int binary_to_octal(long int binary)
{
int octal = 0, decimal = 0, i = 0;

while(binary != 0)
{
decimal += (binary%10) * pow(2,i);
++i;
binary/=10;
}

i = 1;

while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}
return octal;
}

int main()
{
long int binary;

printf("\nEnter a binary number: ");
scanf("%lld", &binary);

printf("\nOctal Equivalent : %d\n", binary_to_octal(binary));
return 0;
}
```

C++

```
// C++ program to convert a number from binary to octal

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

```
int binary_to_octal(long int binary)
{
int octal = 0, decimal = 0, i = 0;

while(binary != 0)
{
decimal += (binary%10) * pow(2,i);
++i;
binary/=10;
}

i = 1;

while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}
return octal;
}

int main()
{
long int binary;

cout << "\nEnter a binary number: ";
cin >> binary;

cout << "\nOctal Equivalent : " << binary_to_octal(binary) << endl;
return 0;
}
```

JAVA

```
// Java program to convert a number from binary to octal

import java.util.*;
public class Main
{
public static int binary_to_octal( int binary)
{
int octal = 0, decimal = 0, i = 0;

while(binary != 0)
{
decimal += (binary%10) * Math.pow(2,i);
++i;
binary/=10;
}

i = 1;
```

```
while (decimal != 0)
{
octal += (decimal % 8) * i;
decimal /= 8;
i *= 10;
}
return octal;
}

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the binary number : ");
int binary = sc.nextInt();
System.out.print("\nEquivalent octal number : " + binary_to_octal(binary));
}
}
```

Python

```
# Python program to convert a number from binary to octal


import math
def binary_to_octal(binary):
octal = 0
decimal = 0
i = 0


while(binary != 0):
decimal += (binary%10) * math.pow(2,i);
i = i + 1
binary = int(binary/10)


i = 1


while (decimal != 0):
octal += (decimal % 8) * i;
decimal = int(decimal/8)
i *= 10;


return octal;


return decimal;


binary = int(input("Enter the binary number : "))
print("Equivalent decimal number : ",int(binary_to_octal(binary)))
```

Problem 72:

Program to convert a number from octal to binary

C

```
// C program to convert a number from octal to binary

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

long long octal_to_binary(int octal)
{
int decimal = 0, i = 0;
long long binary = 0;

while(octal != 0)
{
decimal += (octal%10) * pow(8,i);
++i;
octal/=10;
}

i = 1;

while (decimal != 0)
{
binary += (decimal % 2) * i;
decimal /= 2;
i *= 10;
}

return binary;
}



int main()
{
int octal;

printf("\nEnter an octal number: ");
scanf("%d", &octal);

printf("\nBinary Equivalent : %d\n", octal_to_binary(octal));

return 0;
}
```

C++

```
// C++ program to convert a number from octal to binary
```

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

long long octal_to_binary(int octal)
{
int decimal = 0, i = 0;
long long binary = 0;

while(octal != 0)
{
decimal += (octal%10) * pow(8,i);
++i;
octal/=10;
}

i = 1;

while (decimal != 0)
{
binary += (decimal % 2) * i;
decimal /= 2;
i *= 10;
}

return binary;
}


int main()
{
int octal;

cout << "\nEnter an octal number: ";
cin >> octal;
cout << "\nBinary Equivalent : " << octal_to_binary(octal);

return 0;
}
```

JAVA

```
// Java program to convert a number from octal to binary

import java.util.*;
public class Main
{
public static int octal_to_binary(int octal)
{
int decimal = 0, i = 0;
int binary = 0;
```

```
while(octal != 0)
{
decimal += (octal%10) * Math.pow(8,i);
++i;
octal/=10;
}


i = 1;


while (decimal != 0)
{
binary += (decimal % 2) * i;
decimal /= 2;
i *= 10;
}


return binary;
}

public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the octal number : ");
int octal = sc.nextInt();
System.out.print("\nEquivalent octal number : " + octal_to_binary(octal));
}
}
```

Python

```
# Python program to convert a number from octal to binary


import math
def octal_to_binary(octal):
decimal = 0
i = 0
binary = 0
while(octal != 0):
decimal += (octal%10) * math.pow(8,i)
i = i + 1
octal = int (octal / 10)

i = 1


while (decimal != 0):
binary = binary + (decimal % 2) * i;
decimal = int(decimal / 2);
i *= 10;


return binary


octal = int(input("Enter an octal number : "))
print("Binary Equivalent : ",int(octal_to_binary(octal)))
```

Problem 73:

Program to find prime numbers in a given range using loops

C

// C program to find prime numbers in a given range

```
#include <stdio.h>
int main()
{
int a, b, i, flag;
printf("\nEnter start value : ");
scanf("%d",&a);
printf("\nEnter end value : ");
scanf("%d",&b);
printf("\nPrime Numbers between %d and %d : ", a, b);
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
printf("%d ", a);
++a;
}
printf("\n");
return 0;
}
```

C++

C++ program to find prime numbers in a given range

```
#include <iostream>
using namespace std;
int main()
{
int a, b, i, flag;
cout << "\nEnter start value : ";
cin >> a;
cout << "\nEnter end value : " ;
cin >> b;
cout << "\nPrime Numbers between " << a << " and " << b <<" : ";
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
```

```
flag = 1;
break;
}
}
if (flag == 0)
cout << a << " ";
++a;
}
cout << endl;
return 0;
}
```

JAVA

```
// Java program to find prime numbers in a given range


import java.util.*;
public class Main
{
public static void main(String args[])
{
int a,b,flag,i;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter start value : ");
a = sc.nextInt();
System.out.print("\nEnter end value : ");
b = sc.nextInt();
System.out.print("\nPrime numbers between "+ a + " and " + b + " are : ");
while (a < b)
{
flag = 0;
for(i = 2; i <= a/2; ++i)
{
if(a % i == 0)
{
flag = 1;
break;
}
}
if (flag == 0)
System.out.print(a + " ");
++a;
}
}
}
```

Python

```
# Python program to find prime numbers in a given range


a = int(input("Enter start value : "))
b = int(input("Enter end value : "))
print("Prime numbers between",a,"and",b,"are :")
while (a < b):
flag = 0;
for i in range(2, int(a/2),1):
if(a % i == 0):
```

```
flag = 1;
break
if (flag == 0):
print(a, end = ” “)
a = a + 1
```

Command Line

```
#include<stdio.h>
int main(int argc, char *argv[])
{
int N1, N2, j, i, count, sum = 0;
N1 =atoi(argv[1]);
N2 =atoi(argv[2]);
for(i=N1+1; i<N2; ++i)
{
count = 0;
for(j=2; j<=(i/2); j++)
{
if(i%j==0)
{
count++;
break;
}
}
if(count==0)
sum = sum + i;
}
printf(“%d”,sum);
return 0;
}
```

Problem 74:

Iterative program to reverse a number

C

```
// C program to reverse a number

#include <stdio.h>
int main()
{
int n, rev = 0, rem;
printf("\nEnter a number : ");
scanf("%d", &n);
printf("\nReversed Number : ");
while(n != 0)
{
rem = n%10;
rev = rev*10 + rem;
n /= 10;
}

printf("%d\n", rev);

return 0;
}
```

C++

```
// C++ program to reverse a number

#include <iostream>
using namespace std;

int main()
{
int n, rev = 0, rem;
cout << "\nEnter a number : ";
cin >> n;
cout << "\nReversed Number : ";
while(n != 0)
{
rem = n%10;
rev = rev*10 + rem;
n /= 10;
}

cout << rev << endl;

return 0;
}
```

JAVA

```
// Java program to reverse a number
```

```
import java.util.*;
public class Main
{

public static void main(String[] args)
{
int n, rem = 0, rev = 0;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter a number : ");
n = sc.nextInt();
while(n != 0)
{
rem = n%10;
rev = rev*10 + rem;
n /= 10;
}
System.out.println("Reversed Number : " + rev);



}
}
```

Python

```
# Python program to reverse a number


print("Enter a number : ")
n = int(input())
rev = 0
rem = 0
while(n != 0):
rem = n%10
rev = rev*10 + rem
n = int(n/10)


print("Reversed Number : ",rev)
```

Problem 75:

Recursive program to reverse a number

C

```
// C program to reverse a number using recursion


#include
/* Recursive approach to reverse digits of num*/
int reverse_digits(int num)
{
static int rev_num = 0;
static int base_pos = 1;
if(num > 0)
{
reverse_digits(num/10);
rev_num += (num%10)*base_pos;
base_pos *= 10;
}
return rev_num;
}


int main()
{
int num;
printf("\nEnter a number : ");
scanf("%d",&num);
printf("Reversed number : %d ",reverse_digits(num));
return 0;
}
```

C++

```
// C++ program to reverse a number using recursion


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


/* Recursive approach to reverse digits of num*/
int reverse_digits(int num)
{
static int rev_num = 0;
static int base_pos = 1;
if(num > 0)
{
reverse_digits(num/10);
rev_num += (num%10)*base_pos;
base_pos *= 10;
}
return rev_num;
}


int main()
{
int num;
```

```
cout << "\nEnter a number : ";
cin >> num;
cout << "Reversed number : " << reverse_digits(num);
return 0;
}
```

JAVA

```
// Java program to reverse a number using recursion


class Main
{
static int rev_num = 0;
static int base_pos = 1;


// Recursive approach to reverse digits of num
static int reverse_digits(int num)
{
if(num > 0)
{
reverse_digits(num / 10);
rev_num += (num % 10) * base_pos;
base_pos *= 10;
}
return rev_num;
}


public static void main(String[] args)
{
int num = 1234;
System.out.println(reverse_digits(num));
}
}
```

Python

```
# Python program to reverse a number using recursion


rev_num = 0
base = 1


# Recursive function to reverse a number
def reverse_digits(num):
global rev_num
global base
if(num > 0):
reverse_digits((int)(num / 10))
rev_num += (num % 10) * base
base *= 10
return rev_num


num = 1907
print("Reversed number is ", reverse_digits(num))
```

Problem 76:

Program to reverse a string by swapping the characters using the iterative approach

C

```
/* C program to reverse a string */
#include <stdio.h>


int main()
{
char str[1000], rev[1000];
int i, j, count = 0;


printf("\nEnter the string : ");
gets(str);


while (str[count] != '\0')
count++;


j = count – 1;


for (i = 0; i < count; i++) {
rev[i] = str[j];
j–;
}


rev[i] = '\0';


printf("Reverse of the string : %s\n", rev);


return 0;
}
```

C++

```
// C++ program to reverse a string
#include <bits/stdc++.h>
using namespace std;


void reverse_string(string& str)
{
int n = str.length();



for (int i = 0; i < n / 2; i++)
swap(str[i], str[n – i – 1]);
}


int main()
{
string str;
cout << "\nEnter the string : ";
```

```
cin >> str;
reverse_string(str);
cout << “\nReverse of the string : ” << str << endl;
return 0;
}
```

JAVA

```
/* Java program to reverse a string */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
System.out.println(“Enter the string : “);

Scanner sc = new Scanner(System.in);
String str = sc.nextLine();

StringBuilder sb = new StringBuilder();

for(int i = str.length() – 1; i >= 0; i–)
{
sb.append(str.charAt(i));
}

System.out.println(“Reversed string is:”);
System.out.println(sb.toString());
}
}
```

Python

```
# Python program to reverse a string


str = input(“Enter the string : “)
rev = “”
for i in range(0, len(str)):
rev = rev + str[j]
j = j – 1
print(“Reversed String : “,rev)
```

Problem 77:

Program to reverse a string using recursion

C

```
/* C program to reverse a string using recursion */
#include <stdio.h>
#include <string.h>


void reverse(char *x, int begin, int end)
{
char c;


if (begin >= end)
return;


c = *(x+begin);
*(x+begin) = *(x+end);
*(x+end) = c;


reverse(x, ++begin, –end);
}


int main()
{
char a[100];
printf(“\nEnter the string : “);
gets(a);
reverse(a, 0, strlen(a)-1);
printf(“Reversed string is %s\n”, a);
return 0;
}
```

C++

```
/* C++ program to reverse a string using recursion */
#include <iostream>
#include <string.h>
using namespace std;


void reverse(char *x, int begin, int end)
{
char c;


if (begin >= end)
return;


c = *(x+begin);
*(x+begin) = *(x+end);
*(x+end) = c;


reverse(x, ++begin, –end);
}
```

```
int main()
{
char a[100];
cout << "\nEnter the string : ";
cin >> a;
reverse(a, 0, strlen(a)-1);
cout << "Reversed string is " << a << endl;
return 0;
}
```

JAVA

```
/* Java program to reverse a string using recursion */


import java.util.*;
public class Main
{
public static String reverse_string(String rev)
{
if (rev.isEmpty()){
return rev;
}


rev.substring(1);
return reverse_string(rev.substring(1)) + rev.charAt(0);
}
public static void main(String[] args)
{
System.out.println("Enter the string : ");

Scanner sc = new Scanner(System.in);
String str = sc.nextLine();
System.out.println("Reversed String : " + reverse_string(str));

}
}
```

Problem 78:

Program to reverse a string using standard library function

C++

```
/* C++ program to reverse a string  */
#include <bits/stdc++.h>
using namespace std;


int main()
{
string str;
cout << "\nEnter the string : ";
cin >> str;
reverse(str.begin(), str.end());
cout << "\nReversed string is " << str << endl;
}
```

JAVA

```
/* Java program to reverse a string */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
System.out.println("Enter the string : ");
Scanner sc = new Scanner(System.in);
String str = sc.nextLine();
StringBuilder input = new StringBuilder();
input.append(str);
input = input.reverse();
System.out.println("Reversed String : " + input);
}
}
```

Python

```
# Python program to reverse a string


str = input("Enter the string : ")
s1 = ''.join(reversed(str))
print("Reversed String : ", s1)
```

Problem 79:

Program to print half pyramid pattern using stars

C

```
// C program to print half pyramid pattern using stars
#include <stdio.h>
int main()
{
int i, j,n;
scanf("%d",&n);
for(i = 0; i < n; i++)
{
for(j = 0; j <= i; j++)
{
printf("*");
}
printf("\n");
}
return 0;
}
```

C++

```
// C++ program to print half pyramid pattern using stars
#include <iostream>
using namespace std;


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

JAVA

```
// Java program to print half pyramid pattern using stars
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = 0; i < n; i++)
{
for(j = 0; j <= i; j++)
{
```

```
System.out.print(“*”);
}
System.out.print(“\n”);
}
}
}
```

Python

```
# Python program to print half pyramid pattern using stars
n = int(input())
for i in range(1, n+1) :
for j in range(1, i+1) :
print(“*”, end=””)
print()
```

Problem 80:

Program to print inverted half pyramid pattern using stars

C

```
// C program to print inverted half pyramid pattern using stars
#include <stdio.h>
int main()
{
int i, j, n, k = 0;
scanf("%d",&n);


for(i = n; i >= 1; –i)
{
for(j = 1; j <= i; ++j)
{
printf("* ");
}
printf("\n");
}


return 0;
}
```

C++

```
// C++ program to print inverted half pyramid pattern using stars
#include <iostream>
using namespace std;


int main()
{
int i, j, n, k = 0;
cin >> n;


for(i = n; i >= 1; –i)
{
for(j = 1; j <= i; ++j)
{
cout << "* ";
}
cout << endl;
}


return 0;
}
```

JAVA

```
// Java program to print inverted half pyramid pattern using stars
import java.util.*;
public class Main
{
public static void main(String[] args)
{
```

```
int n, i, j, k = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = n; i >= 1; –i)
{
for(j = 1; j <= i; ++j)
{
System.out.print(“* “);
}
System.out.print(“\n”);
}


}
}
```

Python

```
# Python program to print inverted half pyramid pattern using stars
n = int(input())
for i in range(n,0,-1):
for j in range(1,i+1,1):
print(“* “, end = “”)
print(“\n”)
```

Problem 81:

Program to print full pyramid pattern using stars

C

```
// C program to print full pyramid pattern using stars
#include <stdio.h>
int main()
{
int i, j, n, k = 0;
scanf("%d",&n);


for(i = 1; i <= n; ++i, k = 0)
{
for(j = 1; j <= n – i; ++j)
{
printf(" ");
}


while(k != 2 * i-1)
{
printf("* ");
++k;
}


printf("\n");
}


return 0;
}
```

C++

```
// C++ program to print full pyramid pattern using stars
#include <iostream>
using namespace std;


int main()
{
int i, j, n, k = 0;
cin >> n;


for(i = 1; i <= n; ++i, k = 0)
{
for(j = 1; j <= n – i; ++j)
{
cout << " ";
}


while(k != 2 * i-1)
{
cout << "* ";
++k;
}
```

```
cout << endl;
}

return 0;
}
```

JAVA

```
// Java program to print full pyramid pattern using stars
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j, k = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = 1; i <= n; ++i, k = 0)
{
for(j = 1; j <= n – i; ++j)
{
System.out.print(” “);
}

while(k != 2 * i-1)
{
System.out.print(“* “);
++k;
}

System.out.print(“\n”);
}
}
}
```

Python

```
# Python program to print full pyramid pattern using stars
n = int(input())
for i in range(1,n+1):
k = 0
for j in range(1,n-i+1):
print(” “, end = “”)

while(k != 2 * i-1):
print(“* “, end = “”)
k = k + 1
print(“\n”)
```

Problem 82:

Program to print inverted full pyramid pattern using stars

C

```
// C program to print inverted full pyramid pattern using stars
#include
int main()
{
int i, j, n, k = 0;
scanf(“%d”,&n);


for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
printf(” “);


for(j=i; j <= 2*i-1; ++j)
printf(“* “);


for(j=0; j < i-1; ++j)
printf(“* “);


printf(“\n”);
}


return 0;
}
```

C++

```
// C++ program to print inverted full pyramid pattern using stars
#include <iostream>
using namespace std;


int main()
{
int i, j, n, k = 0;
cin >> n;


for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
{
cout << ” “;
}


for(j=i; j <= 2*i-1; ++j)
{
cout << “* “;
}
```

```
for(j=0; j < i-1; ++j)
{
cout << "* ";
}

cout << endl;
}

return 0;
}
```

JAVA

```
// Java program to print inverted full pyramid pattern using stars
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j, k = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
System.out.print(" ");

for(j=i; j <= 2*i-1; ++j)
System.out.print("* ");

for(j=0; j < i-1; ++j)
System.out.print("* ");

System.out.print("\n");
}


}
}
```

Python

```
# Python program to print inverted full pyramid pattern using stars
n = int(input())
for i in range(n,1,-1):
for j in range(0, n-i+1):
print(" ", end = "")

for j in range(i, 2*i-1,1):
print("* ", end = "")

for j in range(0, i – 1, 1):
print("* ", end ="")

print("\n");
```

Problem 83:

Program to print hollow full pyramid pattern using stars

C

```
#include <stdio.h>
int main()
{
int n;
scanf("%d",&n);
int i, j, k = 0;
for (i = 1; i <= n; i++)
{
for (j = i; j < n; j++) {
printf(" ");
}
while (k != (2 * i – 1)) {
if (k == 0 || k == 2 * i – 2)
printf("*");
else
printf(" ");
k++;
;
}
k = 0;
printf("\n"); // print next row
}
for (i = 0; i < 2 * n – 1; i++) {
printf("*");
}
}
```

C++

```
#include <iostream>
using namespace std;
void printPattern(int);
int main()
{
int n;
cin >> n;
int i, j, k = 0;
for (i = 1; i <= n; i++)
{
for (j = i; j < n; j++) {
cout << " ";
}
while (k != (2 * i – 1)) {
if (k == 0 || k == 2 * i – 2)
cout << "*";
else
cout << " ";
k++;
;
}
k = 0;
cout << endl; // print next row
}
```

```
for (i = 0; i < 2 * n – 1; i++) {
cout << “*”;
}
}
```

JAVA

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
int i, j, k = 0;
for (i = 1; i <= n; i++)
{
for (j = i; j < n; j++) {
System.out.print(” “);
}
while (k != (2 * i – 1)) {
if (k == 0 || k == 2 * i – 2)
System.out.print(“*”);
else
System.out.print(” “);
k++;

}
k = 0;
System.out.print(“\n”); // print next row
}
for (i = 0; i < 2 * n – 1; i++) {
System.out.print(“*”);
}
}


}
```

Problem 84:

Program to print inverted half pyramid pattern

C

```
#include <stdio.h>
int main()
{
int n;
scanf("%d",&n);
int i, j, k = 0;
for (i = 0; i < n; i++)
{
printf("* ");
}
for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
while (k != (2 * i – 1)) {
if (k == 0 || k == 2 * i – 2)
printf("*");
else
printf(" ");
k++;


}
k = 0;
printf("\n"); // print next row
}


}
```

C++

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


int main()
{
int n;
cin >> n;
int i, j, k = 0;
for (i = 0; i < n; i++)
{
cout << "* ";
}
for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
while (k != (2 * i – 1)) {
if (k == 0 || k == 2 * i – 2)
cout << "*";
else
cout << " ";
k++;
```

```
}
k = 0;
cout << endl; // print next row
}


}
```

JAVA

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
int i, j, k = 0;
for (i = 0; i < n; i++)
{
System.out.print("* ");
}
for(i=n; i>=1; –i)
{
for(j=0; j < n-i; ++j)
while (k != (2 * i – 1))
{
if (k == 0 || k == 2 * i – 2)
System.out.print("*");
else
System.out.print(" ");
k++;

}
k = 0;
System.out.print("\n"); // print next row
}
}
}
```

Problem 85:

Program to print inverted hollow full pyramid

C

```
#include <stdio.h>
int main()
{
int n;
scanf("%d", &n);
int i, j;
for (i = 1; i <= n; i++) {
for (j = 1; j < i; j++) {
printf(" ");
}

for (j = 1; j <= (n * 2 – (2 * i – 1)); j++) {

if (i == 1 || j == 1 || j == (n * 2 – (2 * i – 1))) {
printf("*");
} else {
printf(" ");
}
}
printf("\n");
}
}
```

C++

```
#include <iostream>
using namespace std;
int main()
{
int n;
cin >> n;
int i, j;
for (i = 1; i <= n; i++) {
for (j = 1; j < i; j++) {
cout << " ";
}

for (j = 1; j <= (n * 2 – (2 * i – 1)); j++) {

if (i == 1 || j == 1 || j == (n * 2 – (2 * i – 1))) {
cout << "*";
} else {
cout << " ";
}
}
cout << endl;
}
}
```

JAVA

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
int i, j;
for (i = 1; i <= n; i++) {
for (j = 1; j < i; j++) {
System.out.print(” “);
}


for (j = 1; j <= (n * 2 – (2 * i – 1)); j++) {


if (i == 1 || j == 1 || j == (n * 2 – (2 * i – 1))) {
System.out.print(“*”);
} else {
System.out.print(” “);
}
}
System.out.print(“\n”);
}
}
}
```

Problem 86:

Program to print half pyramid pattern using numbers

C

```
/* C program – Half Pyramid Pattern Printing using numbers */
#include
int main()
{
int i, j,n;
scanf("%d",&n);
for(i = 1; i <= n; i++)
{
for(j = 1; j <= i; j++)
{
printf("%d ",j);
}
printf("\n");
}
return 0;
}
```

C++

```
// C++ program – Half Pyramid Pattern Printing using numbers
#include <iostream>
using namespace std;


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

JAVA

```
/* Java program – Half Pyramid Pattern Printing using numbers */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
int i,j;
for(i = 1; i<= n; i++)
```

```
{
for(j = 1; j <= i; j++)
{
System.out.print(j + ” “);
}
System.out.println();
}
}
}
```

Python

```
# Python program – Half Pyramid Pattern Printing using numbers
n = int(input())
for i in range(1,n+1):
for j in range(1,i+1):
print(j, end = ” “)
print()
```

Problem 87:

Program to print inverted half pyramid pattern printing using numbers

C

```
/* C program – Pyramid Pattern Printing using numbers */
#include
int main()
{
int i, j,n;
scanf("%d",&n);
for(i = 1; i <= n; i++)
{
for(j = i; j <= n; j++)
{
printf("%d ",j);
}
printf("\n");
}
return 0;
}
```

C++

```
// C++ program – Pyramid Pattern Printing using numbers
#include <iostream>
using namespace std;


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

JAVA

```
/* Java program – Pyramid Pattern Printing using numbers */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
int i,j;
for(i = 1; i<= n; i++)
```

```
{
for(j = i; j <= n; j++)
{
System.out.print(j + ” “);
}
System.out.println();
}
}
}
```

Python

```
# Python program – Pyramid Pattern Printing using numbers
n = int(input())
for i in range(1,n+1):
for j in range(i,n+1):
print(j, end = ” “)
print()
```

Problem 88:

Program to print full pyramid using numbers

C

```
/* C program – Full Pyramid Pattern Printing using numbers */
#include
int main()
{
int i, j, n, count = 0, count1 = 0, k = 0;
scanf("%d",&n);
for(i = 1; i <= n; ++i)
{
for(j = 1; j <= n-i; ++j)
{
printf(" ");
++count;
}

while(k != 2*i-1)
{
if (count <= n-1)
{
printf("%d ", i+k);
++count;
}
else
{
++count1;
printf("%d ", (i+k-2*count1));
}
++k;
}
count1 = count = k = 0;

printf("\n");
}
return 0;
}
```

C++

```
/* C program – Full Pyramid Pattern Printing using numbers */
#include
int main()
{
int i, j, n, count = 0, count1 = 0, k = 0;
scanf("%d",&n);
for(i = 1; i <= n; ++i)
{
for(j = 1; j <= n-i; ++j)
{
printf(" ");
++count;
}
```

```
while(k != 2*i-1)
{
if (count <= n-1)
{
printf("%d ", i+k);
++count;
}
else
{
++count1;
printf("%d ", (i+k-2*count1));
}
++k;
}
count1 = count = k = 0;


printf("\n");
}
return 0;
}
```

JAVA

```
/* Java program – Full Pyramid Pattern Printing using numbers */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
int i, j, n, count = 0, count1 = 0, k = 0;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();


for(i = 1; i <= n; ++i)
{
for(j = 1; j <= n-i; ++j)
{
System.out.print(" ");
++count;
}


while(k != 2*i-1)
{
if (count <= n-1)
{
System.out.print(i+k + " ");
++count;
}
else
{
++count1;
System.out.print((i+k-2*count1) + " ");
}
++k;
```

```
}
count1 = count = k = 0;


System.out.println();
}
}
}
```

Python

```
# Python program – Full Pyramid Pattern Printing using numbers
n = int(input())
count = count1 = k = 0
for i in range(1,n+1):
for j in range(1,n-i+1):
print(” “, end = ” “)
count = count + 1
while(k != 2*i-1):
if (count <= n-1):
print(i+k, end = ” “)
count = count + 1
else:
count1 = count1 + 1
print((i+k-2*count1), end = ” “)
k = k + 1
count1 = count = k = 0


print()
```

Problem 89:

Program to print hollow half pyramid pattern using numbers

C

```
/* C program print hollow half pyramid pattern using numbers */
#include <stdio.h>
int main()
{
int i, j, n, k = 0;
scanf("%d",&n);
for(i = 1; i <= n; i++)
{
for(j = 1; j <= i; j++)
{
if (j == 1 || j == i || i == n)
printf("%d", j);
else
printf(" ");
}
printf("\n");
}
return 0;
}
```

C++

```
/* C++ program print hollow half pyramid pattern using numbers */
#include <iostream>
using namespace std;


int main()
{
int i, j, n, k = 0;
cin >> n;
for(i = 1; i <= n; i++)
{
for(j = 1; j <= i; j++)
{
if (j == 1 || j == i || i == n)
cout << j;
else
cout << " ";
}
cout << endl;
}
return 0;
}
```

JAVA

```
/* Java program print hollow half pyramid pattern using numbers */
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j;
```

```
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = 1; i <= n; i++)
{
for(j = 1; j <= i; j++)
{
if (j == 1 || j == i || i == n)
System.out.print(j);
else
System.out.print(” “);
}
System.out.println();
}
}
}
```

Python

```
# Python program print hollow half pyramid pattern using numbers */
n = 5
for i in range(1, n+1):
for j in range(1, i+1):
if(j == 1 or j == i or i == n):
print(j, end = “”)
else:
print(” “, end = “”)
print()
```

Problem 90:

Program to print hollow inverted half pyramid pattern using numbers

C

```
/* C program print hollow inverted half pyramid pattern using numbers */
#include <stdio.h>
int main()
{
int i, j, n, k = 0;
scanf("%d",&n);
for(i = 1; i <= n; i++)
{
for(j = i; j <= n; j++)
{
if (i == 1 || j == i || j == n)
printf("%d", j);
else
printf(" ");
}
printf("\n");
}
return 0;
}
```

C++

```
/* C++ program to print hollow inverted half pyramid pattern using numbers */
#include <iostream>
using namespace std;


int main()
{
int i, j, n, k = 0;
cin >> n;
for(i = 1; i <= n; i++)
{
for(j = i; j <= n; j++)
{
if (i == 1 || j == i || j == n)
cout << j;
else
cout << " ";
}
cout << endl;
}
return 0;
}
```

JAVA

```
/* Java program print hollow half pyramid pattern using numbers */
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j;
```

```
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = 1; i <= n; i++)
{
for(j = i; j <= n; j++)
{
if (i == 1 || j == n || j == i)
System.out.print(j);
else
System.out.print(” “);
}
System.out.println();
}
}
}
```

Python

```
# Python program to print inverted hollow half pyramid pattern using numbers
n = int(input())
for i in range(1, n+1):
for j in range(i, n+1):
if(i == 1 or j == i or j == n):
print(j, end = “”)
else:
print(” “, end = “”)
print()
```

Problem 91:

Program to print hollow inverted full pyramid pattern using numbers

C

```
/* C program to print hollow full pyramid pattern using numbers */
#include <stdio.h>
int main()
{
int i, j, n;
printf("Enter value of n : ");
scanf("%d", &n);
for(i = 1; i <= n; i++)
{
for(j = i; j < n; j++)
{
printf(" ");
}
for(j = 1; j <= i; j++)
{
if(j == 1 || i == n)
{
printf("%d ", j);
}
else
{
printf(" ");
}
}
for(j = 1; j < i; j++)
{
if(j == i-1 && j < n-1)
{
printf("%d", j+1);
}
else
{
printf(" ");
}
}
printf("\n");
}
return 0;
}
```

C++

```
/* C++ program to print hollow full pyramid pattern using numbers */
#include <iostream>
using namespace std;


int main()
{
int i, j, n;
cout << "Enter value of n : ";
cin >> n;
for(i = 1; i <= n; i++)
```

```
{
for(j = i; j < n; j++)
{
cout << " ";
}
for(j = 1; j <= i; j++)
{
if(j == 1 || i == n)
{
cout << j << " ";
}
else
{
cout << " ";
}
}
for(j = 1; j < i; j++)
{
if(j == i-1 && j < n-1)
{
cout << j+1;
}
else
{
cout << " ";
}
}
cout << endl;
}
return 0;
}
```

JAVA

```
/* Java program print hollow full pyramid pattern using numbers */
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n, i, j;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
for(i = 1; i <= n; i++)
{
for(j = i; j < n; j++)
{
System.out.print(" ");
}
for(j = 1; j <= i; j++)
{
if(j == 1 || i == n)
{
System.out.print(j + " ");
}
else
{
```

```
System.out.print(” “);
}
}
for(j = 1; j < i; j++)
{
if(j == i-1 && j < n-1)
{
System.out.print(j+1);
}
else
{
System.out.print(” “);
}
}
System.out.print(“\n”);
}
}
}
```

Problem 92:

Program for solid diamond pattern using stars

C

```
/* C program – solid diamond pattern printing using stars */
#include <stdio.h>


int main()
{
int n, c, k, space = 1;


printf(“Enter the number of rows\n”);
scanf(“%d”, &n);


space = n – 1;


for (k = 1; k <= n; k++)
{
for (c = 1; c <= space; c++)
printf(” “);


space–;


for (c = 1; c <= 2*k-1; c++)
printf(“*”);


printf(“\n”);
}


space = 1;


for (k = 1; k <= n – 1; k++)
{
for (c = 1; c <= space; c++)
printf(” “);


space++;


for (c = 1 ; c <= 2*(n-k)-1; c++)
printf(“*”);


printf(“\n”);
}


return 0;
}
```

C++

```
/* C++ program – solid diamond pattern printing using stars */
#include <iostream>
using namespace std;

int main()
{
int n, c, k, space = 1;

cout << "\nEnter the number of rows : ";
cin >> n;

space = n – 1;

for (k = 1; k <= n; k++)
{
for (c = 1; c <= space; c++)
cout << " ";

space–;

for (c = 1; c <= 2*k-1; c++)
cout << "*";

cout << endl;
}

space = 1;

for (k = 1; k <= n – 1; k++)
{
for (c = 1; c <= space; c++)
cout << " ";

space++;

for (c = 1 ; c <= 2*(n-k)-1; c++)
cout << "*";

cout << endl;
}

return 0;
}
```

JAVA

```
/* Java program – solid diamond pattern printing using stars */
import java.util.*;
```

```
public class Main
{

public static void main(String[] args)
{

int n, c, k, space = 1;
System.out.print("\nEnter the number of rows : ");
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
System.out.println();
space = n – 1;

for (k = 1; k <= n; k++)
{
for (c = 1; c <= space; c++)
System.out.print(" ");

space–;

for (c = 1; c <= 2*k-1; c++)
System.out.print("*");

System.out.print("\n");
}

space = 1;

for (k = 1; k <= n – 1; k++)
{
for (c = 1; c <= space; c++)
System.out.print(" ");

space++;

for (c = 1 ; c <= 2*(n-k)-1; c++)
System.out.print("*");

System.out.print("\n");
}

}
}
```

Python

```
# Python program – solid diamond pattern printing using stars
def Diamond_pattern(rows):
n = 0
for i in range(1, rows + 1):
for j in range (1, (rows – i) + 1):
print(end = " ")

while n != (2 * i – 1):
print("*", end = "")
n = n + 1
```

```
n = 0

print()


k = 1
n = 1
for i in range(1, rows):
for j in range (1, k + 1):
print(end = ” “)
k = k + 1

while n <= (2 * (rows – i) – 1):
print(“*”, end = “”)
n = n + 1
n = 1
print()


rows = int(input(“Enter the number of rows : “))
Diamond_pattern(rows)
```

Problem 93:

Program for hollow diamond pattern using stars

C

```
/* C program – hollow diamond pattern printing using stars */


#include <stdio.h>


int main()
{
int i, j, space, k = 0, n;
printf("\nEnter the number of rows : ");
scanf("%d",&n);
for (i = 1; i <= n; i++)
{
for (j = 1; j <= n – i; j++)
{
printf(" ");
}
while (k != (2 * i – 1))
{
if (k == 0 or k == 2 * i – 2)
printf("*");
else
printf(" ");
k++;
}
k = 0;
printf("\n");
}
n–;
for (i = n; i >= 1; i–)
{
for (j = 0; j <= n – i; j++)
{
printf(" ");
}
k = 0;
while (k != (2 * i – 1))
{
if (k == 0 or k == 2 * i – 2)
printf("*");
else
printf(" ");
k++;
}
printf("\n");
}
}
```

C++

```
/* C++ program – hollow diamond pattern printing using stars */
```

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


int main()
{
int i, j, space, k = 0, n;
cout << "\nEnter the number of rows : ";
cin >> n;
for (i = 1; i <= n; i++)
{
for (j = 1; j <= n – i; j++)
{
cout << " ";
}
while (k != (2 * i – 1))
{
if (k == 0 or k == 2 * i – 2)
cout << "*";
else
cout << " ";
k++;
}
k = 0;
cout << endl;
}
n–;
for (i = n; i >= 1; i–)
{
for (j = 0; j <= n – i; j++)
{
cout << " ";
}
k = 0;
while (k != (2 * i – 1))
{
if (k == 0 or k == 2 * i – 2)
cout << "*";
else
cout << " ";
k++;
}
cout << endl;
}
}
```

JAVA

```
/* Java program – hollow diamond pattern printing using stars */
import java.util.*;


public class Main
{


public static void main(String[] args)
{
```

```
int i, j, space, k = 0, n;
System.out.print("\nEnter the number of rows : ");
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
System.out.println();


for (i = 1; i <= n; i++)
{
for (j = 1; j <= n – i; j++)
{
System.out.print(" ");
}
while (k != (2 * i – 1))
{
if ((k == 0) || (k == 2 * i – 2))
System.out.print("*");
else
System.out.print(" ");
k++;
}
k = 0;
System.out.print("\n");
}
n–;
for (i = n; i >= 1; i–)
{
for (j = 0; j <= n – i; j++)
{
System.out.print(" ");
}
k = 0;
while (k != (2 * i – 1))
{
if ((k == 0) || (k == 2 * i – 2))
System.out.print("*");
else
System.out.print(" ");
k++;
}
System.out.print("\n");
}
}
}
```

Python

```
# Python program – hollow diamond pattern printing using stars


def Diamond_pattern(n) :
k = 0;


for i in range(1,n+1) :


for j in range(1,n-i+1) :
print(" ",end="")
```

```
while (k != (2 * i – 1)) :
if (k == 0 or k == 2 * i – 2) :
print(“*”,end=””)
else :
print(” “,end=””)
k = k + 1

k = 0


print(“”),

n = n – 1


for i in range (n,0,-1) :
for j in range(0,n-i+1) :
print(” “,end=””)

k = 0
while (k != (2 * i – 1)) :
if (k == 0 or k == 2 * i – 2) :
print(“*”,end=””)
else :
print(” “,end=””)
k = k + 1

print(“”),

n = int(input(“Enter the number of rows : “))
Diamond_pattern(n)
```

Problem 94:

Program for solid half diamond pattern printing using stars

C

```
/* C program – solid half diamond pattern printing using stars */

#include

int main()
{
int i, j, space, k = 0, n;
printf("\nEnter the number of rows : ");
scanf("%d",&n);
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n-i;j++)
{
printf(" ");
}
for(int j=1;j<=i;j++)
{
printf("*");
}

printf("\n");
}

for(int i=n-1;i>0;i–)
{
for(int j=1;j<=n-i;j++)
{
printf(" ");
}
for(int j=1;j<=i;j++)
{
printf("*");
}
printf("\n");
}
}
```

C++

```
/* C++ program – solid half diamond pattern printing using stars */

#include <iostream>
using namespace std;

int main()
{
int i, j, space, k = 0, n;
cout << "\nEnter the number of rows : ";
cin >> n;
```

```
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n-i;j++)
{
cout << " ";
}
for(int j=1;j<=i;j++)
{
cout << "*";
}

cout << endl;
}

for(int i=n-1;i>0;i–)
{
for(int j=1;j<=n-i;j++)
{
cout << " ";
}
for(int j=1;j<=i;j++)
{
cout << "*";
}
cout << endl;
}
}
```

JAVA

```
/* Java program – solid half diamond pattern printing using stars */
import java.util.*;

public class Main
{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter N : ");
int n=sc.nextInt();
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n-i;j++)
{
System.out.print(" ");
}
for(int j=1;j<=i;j++)
{
System.out.print("*");
}

System.out.println();
}

for(int i=n-1;i>0;i–)
{
```

```
for(int j=1;j<=n-i;j++)
{
System.out.print(” “);
}
for(int j=1;j<=i;j++)
{
System.out.print(“*”);
}
System.out.println();
}
}
}
```

Problem 95:

Program for palindromic pattern printing in a half diamond

C

```
#include <stdio.h>


int main()
{
int i, j, N;


printf(“Enter the number of rows : “);
scanf(“%d”, &N);
printf(“*\n”);


for(i=1; i<=N; i++)
{
for(j=1; j<=i; j++)
{
if(j == 1)
printf(“*”);
printf(“%d”, j);
}


for(j=i-1; j>=1; j–)
{
printf(“%d”, j);
}
printf(“*”);
printf(“\n”);
}


for(i=N-1; i>=1; i–)
{
for(j=1; j<=i; j++)
{
if(j == 1)
printf(“*”);
printf(“%d”, j);
}


for(j=i-1; j>=1; j–)
{
printf(“%d”, j);
}
printf(“*”);


printf(“\n”);
}
printf(“*\n”);
return 0;
}
```

C++

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

int main()
{
int i, j, N;

cout << “Enter the number of rows : “;
cin >> N;
cout << “*\n”;

for(i=1; i<=N; i++)
{
for(j=1; j<=i; j++)
{
if(j == 1)
cout << “*”;
cout << j;
}

for(j=i-1; j>=1; j–)
{
cout << j;
}
cout << “*”;
cout << “\n”;
}

for(i=N-1; i>=1; i–)
{
for(j=1; j<=i; j++)
{
if(j == 1)
cout << “*”;
cout << j;
}

for(j=i-1; j>=1; j–)
{
cout << j ;
}
cout << “*”;

cout << “\n”;
}
cout << “*\n”;
return 0;
}
```

JAVA

```
/* Java program for Palindrome pyramid pattern printing using numbers */
import java.util.*;
public class Main {


public static void main(String[] args) {
int N, i, j;
Scanner sc = new Scanner(System.in);
System.out.print("\nEnter the number of rows : ");
System.out.println();
N = sc.nextInt();
System.out.print("*\n");
for(i=1; i<=N; i++)
{
for(j=1; j<=i; j++)
{
if(j == 1)
System.out.print("*");
System.out.print(j);
}


for(j=i-1; j>=1; j–)
{
System.out.print(j);
}
System.out.print("*");
System.out.print("\n");
}


for(i=N-1; i>=1; i–)
{
for(j=1; j<=i; j++)
{
if(j == 1)
System.out.print("*");
System.out.print(j);
}


for(j=i-1; j>=1; j–)
{
System.out.print(j);
}
System.out.print("*");


System.out.print("\n");
}
System.out.print("*\n");
}
}
```

Python

```
N = int(input("\nEnter the number of rows : "))
print()
```

```
print(“*”)
for i in range(1,N+1):
for j in range(1, i+1):
if(j == 1):
print(“*”, end = “”)
print(j,end = “”)


for j in range(i-1, 0, -1):
print(j,end = “”)
print(“*”)


for i in range(N-1, 0, -1):


for j in range(1, i+1):

if(j == 1):
print(“*”,end = “”)
print(j,end = “”)


for j in range(i-1, 0, -1):
print(j,end = “”)
print(“*”)


print(“*\n”);
```

Problem 96:

Program for half diamond pattern printing using numbers

C

```
/* C program for diamond pattern printing using numbers */


#include <stdio.h>
int main()
{
int i,j,s,N,count=0;
scanf("%d%d",&s,&N);
for(i=s;count<4;count++)
{
for(j=0;j<count+1;j++)
printf("%d",i);
printf("\n");
i=i+1;
}
for(i=s+N-2;count>0;count–)
{
for(j=0;j<count-1;j++)
printf("%d",i);
printf("\n");
i=i-1;
}
return 0;
}
```

C++

```
/* C++ program for diamond pattern printing using numbers */


#include <iostream>
using namespace std;


int main()
{
int i,j,s,N,count=0;
cin >> s >> N;
for(i=s;count<4;count++)
{
for(j=0;j<count+1;j++)
cout << i;
cout << endl;
i=i+1;
}
for(i=s+N-2;count>0;count–)
{
for(j=0;j<count-1;j++)
cout << i;
cout << endl;
i=i-1;
}
return 0;
}
```

JAVA

```
/* Java program for diamond pattern printing using numbers */


import java.util.*;


public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);


int i,j,s,N,count=0;
s = sc.nextInt();
N = sc.nextInt();
for(i=s;count<4;count++)
{
for(j=0;j<count+1;j++)
System.out.print(i);
System.out.print("\n");
i=i+1;
}
for(i=s+N-2;count>0;count–)
{
for(j=0;j<count-1;j++)
System.out.print(i);
System.out.print("\n");
i=i-1;
}


}
}
```

Problem 97:

Program for half diamond pattern printing using numbers and stars

C

```
/* C program for diamond pattern printing using  numbers */


#include <stdio.h>
int main()
{
int i,j,k,N,count=0;
scanf("%d",&N);
for(i=1;i<=N;i++)
{
k=1;
for(j=0;j<i;j++)
{
printf("%d",i);
if(k<i)
{
printf("*");
k=k+1;
}
}
printf("\n");
}
for(i=N;i>0;i–)
{
k=1;
for(j=0;j<i;j++)
{
printf("%d",i);
if(k<i)
{
printf("*");
k=k+1;
}
}
printf("\n");
}
return 0;
}
```

C++

```
/* C+ program for diamond pattern printing using numbers and stars */


#include <iostream>
using namespace std;


int main()
{
int i,j,k,N,count=0;
cin >> N;
for(i=1;i<=N;i++)
{
```

```
k=1;
for(j=0;j<i;j++)
{
cout << i;
if(k<i)
{
cout << “*”;
k=k+1;
}
}
cout << endl;
}
for(i=N;i>0;i–)
{
k=1;
for(j=0;j<i;j++)
{
cout << i;
if(k<i)
{
cout << “*”;
k=k+1;
}
}
cout << endl;
}
return 0;
}
```

JAVA

```
/* Java program for diamond pattern printing using numbers and stars */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);


int i,j,k,N,count=0;
N = sc.nextInt();
for(i=1;i<=N;i++)
{
k=1;
for(j=0;j<i;j++)
{
System.out.print(i);
if(k<i)
{
System.out.print(“*”);
k=k+1;
}
}
System.out.print(“\n”);
}
```

```
for(i=N;i>0;i–)
{
k=1;
for(j=0;j<i;j++)
{
System.out.print(i);
if(k<i)
{
System.out.print(“*”);
k=k+1;
}
}
System.out.println();
}


}
}
```

Problem 98:

Program to print the pattern using numbers and stars

C

```
/* C program for diamond pattern printing using numbers and stars */


#include <stdio.h>
int main()
{
int i,j,count=1,n;
printf("Enter a number\n");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
{
if(j<i)
printf("%d*",count++);
else
printf("%d",count++);
}
printf("\n");
}
count=count-n;
for(i=n;i>=1;i–)
{ for(j=1;j<=i;j++)
{
if(j<i)
printf("%d*",count++);
else
printf("%d",count++);
}
count=(count+1)-2*i;
printf("\n");
}
return 0;
}
```

C++

```
/* C++ program for diamond pattern printing using numbers and stars */


#include <iostream>
using namespace std;


int main()
{
int i,j,count=1,n;
cout << "Enter a number\n";
cin >> n;
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
{
```

```
if(j<i)
cout << count++ << “*”;
else
cout << count++;
}
cout << endl;
}
count=count-n;
for(i=n;i>=1;i–)
{ for(j=1;j<=i;j++)
{
if(j<i)
cout << count++ << “*”;
else
cout << count++;
}
count=(count+1)-2*i;
cout << endl;
}
return 0;
}
```

JAVA

```
/* Java program for diamond pattern printing using numbers and stars */


import java.util.*;
public class Main
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);


int i,j,count=1,n;
n = sc.nextInt();
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
{
if(j<i)
System.out.print(count++ + “*”);
else
System.out.print(count++);
}
System.out.print(“\n”);
}
count=count-n;
for(i=n;i>=1;i–)
{ for(j=1;j<=i;j++)
{
if(j<i)
System.out.print(count++ + “*”);
else
System.out.print(count++);
}
```

```
count=(count+1)-2*i;
System.out.println();
}


}
}
```

Problem 99:

To find the second smallest element in an array

C

```
#include <stdio.h>
void sort(int arr[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)
{
for (j = 0; j < n-i-1; j++)
{
if (arr[j] > arr[j+1])
{
int temp = arr[j];
arr[j] = arr[j+1];
arr[j+1] = temp;
}
}
}
}

int main()
{
int n,i;
printf("\nEnter the number of elements : ");
scanf("%d",&n);
int arr[n];
printf("\nInput the array elements : ");
for(i = 0; i < n; i++)
{
scanf("%d",&arr[i]);
}
sort(arr, n);
printf("\nThe second smallest element is %d \n",arr[1]);
return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
int n,i;
cout << "Enter the number of elements : ";
cin >> n;
int arr[n];
cout << "\nInput the array elements : ";
for(i = 0; i < n; i++)
{
cin >> arr[i];
```

```
}
sort(arr, arr+n);
cout << "\nThe second largest element is " << arr[1];
cout << endl;
return 0;
}
```

Problem 100:

To find the second smallest element in an array

C

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

int main()
{
int n,i;
printf("\nEnter the number of elements : ");
scanf("%d",&n);
int arr[n];
printf("\nInput the array elements : ");
for(i = 0; i < n; i++)
{
scanf("%d",&arr[i]);
}
int min = INT_MAX;
int second_min = INT_MAX ;
if (n < 2)
{
printf("\nInvalid Input");
}
for(i = 0; i < n; i++)
{
if(arr[i] < min)
{
second_min = min;
min = arr[i];
}
}
for(i = 0; i < n; i++)
{
if(arr[i] < second_min && arr[i] != min)
{
second_min = arr[i];
}
}


printf("\nThe second smallest element is %d \n", second_min);
return 0;
}
```

OR

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
```

```
int main()
{
int n,i;
printf("\nEnter the number of elements : ");
scanf("%d",&n);
int arr[n];
printf("\nInput the array elements : ");
for(i = 0; i < n; i++)
{
scanf("%d",&arr[i]);
}
int min = INT_MAX;
int second_min = INT_MAX ;
if (n < 2)
{
printf("\nInvalid Input");
}
for(i = 0; i < n; i++)
{
if(arr[i] < min)
{
second_min = min;
min = arr[i];
}
else if(arr[i] < second_min && arr[i] != min)
{
second_min = arr[i];
}
}
printf("\nThe second smallest element is %d \n", second_min);
return 0;
}
```

C++

```
#include<iostream>
using namespace std;
int main()
{
int min = INT_MAX;
int i,n = 5;
cout << "\nEnter the number of elements : ";
cin >> n;
int arr[n];
cout << "\nInput the array elements : ";
for(i = 0; i < n; i++)
{
cin >> arr[i];
}
int second_min = INT_MAX ;
if (n < 2)
{
cout <<"\nInvalid Input";
}
for(i = 0; i < n; i++)
```

```
{
if(arr[i] < min)
{
second_min = min;
min = arr[i];
}
}
for(i=0;i<n;i++)
{
if(arr[i] < second_min && arr[i] != min)
{
second_min = arr[i];
}
}
cout << "\nThe second smallest element is " << second_min;
cout << endl;
return 0;
}
```

OR

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

int main()
{
    int n,i;
    cout << "\nEnter the number of elements :  ";
    cin >> n;
    int arr[n];
    cout << "\nInput the array elements : ";
    for(i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    int min = INT_MAX;
    int second_min = INT_MAX ;
    if (n < 2)
    {
        cout << "\nInvalid Input";
    }
    for(i = 0; i < n; i++)
    {
        if(arr[i] < min)
        {
            second_min = min;
            min = arr[i];
        }
        else if(arr[i] < second_min && arr[i] != min)
        {
            second_min = arr[i];
        }
    }
    cout << "\nThe second smallest element is "<< second_min;
```

```
    cout << endl;
    return 0;
}
```

Problem 101:

Program to remove duplicate elements in an array (sorted array)

C

```
/* C program to remove duplicate elements in an array */
#include<stdio.h>


int remove_duplicate_elements(int arr[], int n)
{


if (n==0 || n==1)
return n;


int temp[n];


int j = 0;
int i;
for (i=0; i<n-1; i++)
if (arr[i] != arr[i+1])
temp[j++] = arr[i];
temp[j++] = arr[n-1];


for (i=0; i<j; i++)
arr[i] = temp[i];


return j;
}



int main()
{
int n;
scanf("%d",&n);
int arr[n];
int i;
for(i = 0; i < n; i++)
{
scanf("%d",&arr[i]);
}


n = remove_duplicate_elements(arr, n);



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


return 0;
}
```

C++

```
/* C++ program to remove duplicate elements in an array */
#include<iostream>
using namespace std;

int remove_duplicate_elements(int arr[], int n)
{

if (n==0 || n==1)
return n;

int temp[n];

int j = 0;
int i;
for (i=0; i<n-1; i++)
if (arr[i] != arr[i+1])
temp[j++] = arr[i];
temp[j++] = arr[n-1];

for (i=0; i<j; i++)
arr[i] = temp[i];

return j;
}



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

n = remove_duplicate_elements(arr, n);



for (i=0; i<n; i++)
cout << arr[i] << ” “;

return 0;
}
```

JAVA

```
/* Java program to remove duplicate elements in an array */
```

```
import java.util.Scanner;
public class Main
{
public static int remove_duplicate_elements(int arr[], int n)
{

if (n==0 || n==1)
return n;

int temp[] = new int[n];

int j = 0;
int i;
for (i=0; i<n-1; i++)
if (arr[i] != arr[i+1])
temp[j++] = arr[i];
temp[j++] = arr[n-1];

for (i=0; i<j; i++)
arr[i] = temp[i];

return j;
}

public static void main(String[] args)
{
int n;
Scanner s = new Scanner(System.in);
System.out.print("Enter no. of elements you want in array:");
n = s.nextInt();
int a[] = new int[n+1];
System.out.println("Enter all the elements:");

for(int i = 0; i < n; i++)
{
a[i] = s.nextInt();
}

n = remove_duplicate_elements(a, n);
System.out.print("Array after removing : ");
for (int i=0; i<n; i++)
System.out.print(a[i] + " ");

}
}
```

Python

```
# Python program to remove duplicate elements in an array



def remove_duplicate_elements(arr, n):


if n == 0 or n == 1:
return n
```

```
temp = list(range(n))

j = 0;
for i in range(0, n-1):

if arr[i] != arr[i+1]:
temp[j] = arr[i]
j += 1

temp[j] = arr[n-1]
j += 1

for i in range(0, j):
arr[i] = temp[i]

return j



n = int(input())
sum = 0
arr = []
for i in range(0,n):
temp = int(input())
arr.append(temp)
n = remove_duplicate_elements(arr,n)
print("Array after removing : ",end = "")
for i in range(0,n):
print(arr[i], end = " ")
```

C (using Pointers)

```
#include<stdio.h>
#include<stdlib.h>
int i,j;
int removed(int*,int);
int main()
{
int n,*a;
scanf("%d",&n);
a=(int*)malloc(n*sizeof(int));
for(i=0;i<n;i++)
scanf("%d",(a+i));

removed(a,n);

return 0;
}
int removed(int*a,int n)
{
int k;
for(i=0;i<n;i++)
{
for(j=i+1;j<n;)
{
if(*(a+i)==*(a+j))
```

```
{
for(k=j;k<n;k++)
{
*(a+k)==*(a+k+1);
}
n–;
}
else
j++;
}
}
for(i=0;i<n;i++)
printf(“%d\n”,*(a+i));
return 0;
}
```

Problem 102:

Algorithm to remove duplicate elements in an array (unsorted array)

C++

```
// C++ program to remove the duplicateelements in an array
#include <iostream>
#include <unordered_map>
using namespace std;


void remove_duplicate_elements(int arr[], int n)
{
// Hash map which will store the elements which has appeared previously.
unordered_map<int, bool> mp;


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


// Print the element if it is there in the hash map
if (mp.find(arr[i]) == mp.end()) {
cout << arr[i] << " ";
}


// Insert the element in the hash map
mp[arr[i]] = true;
}
}


int main(int argc, char const* argv[])
{
int n;
cin >> n;
int arr[n];
int i;
for(i = 0; i < n; i++)
{
cin >> arr[i];
}
remove_duplicate_elements(arr, n);
return 0;
}
```