Coding Question-1

Considerthe peiow 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 Otherthan 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

Solution –

```
#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);
}
```

```
Please comment the code in other languages as well.
```

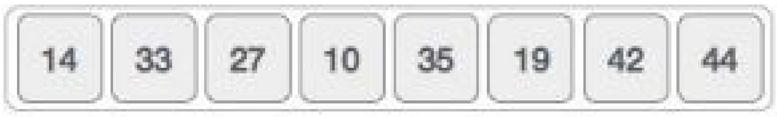
Coding Question-2

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.

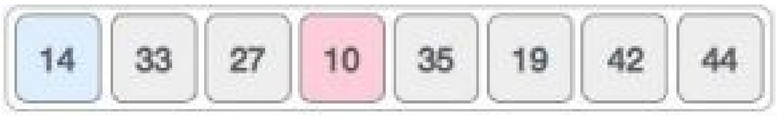
The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

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.'

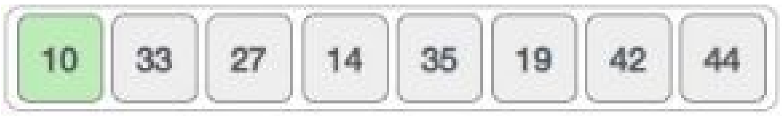
Consider the following depicted array as an example.



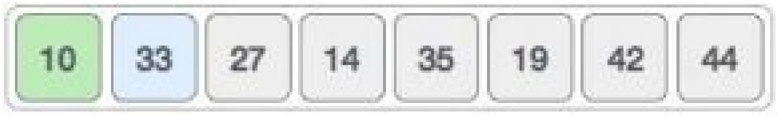
For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.



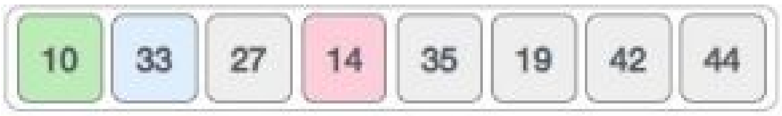
Click edit button to change this text. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut elit tellus, luctus nec ullamcorper mattis, pulvinar dapibus leo.



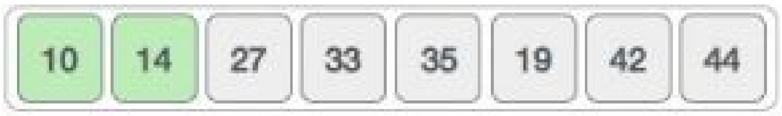
Click edit button to change this text. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut elit tellus, luctus nec ullamcorper mattis, pulvinar dapibus leo.



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 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;
}
```

In 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]),
```

In 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);
}
}
```

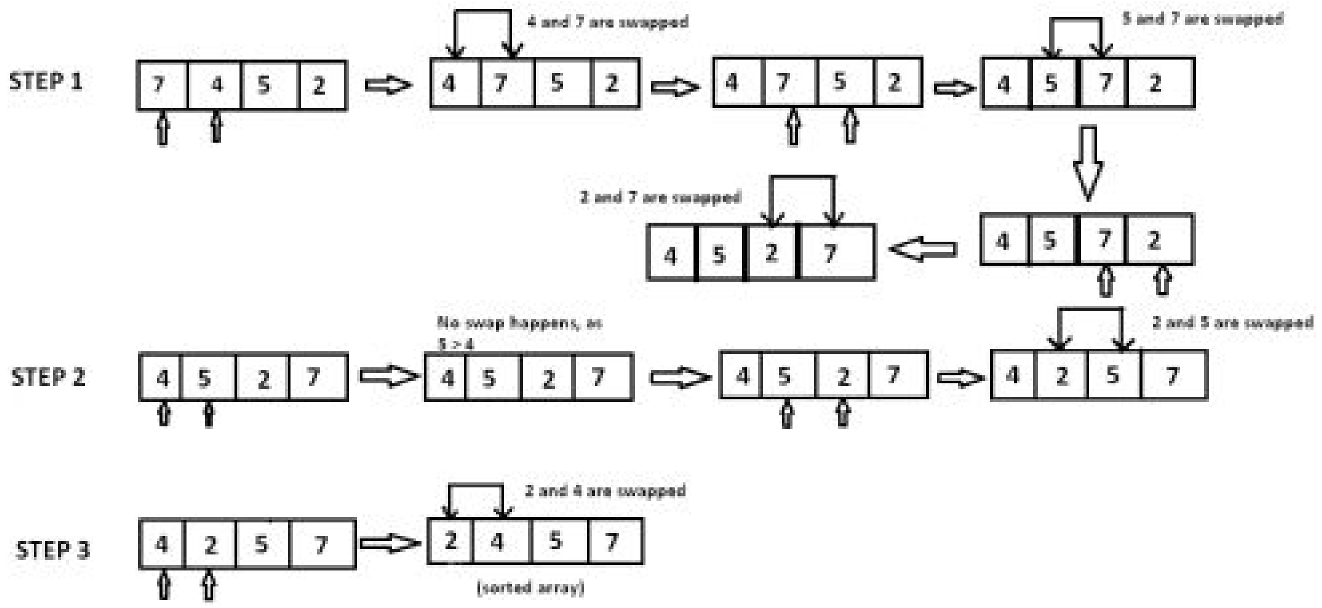
Coding Question-3

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]),
```

Coding Question-4

Longest Common Subsequence –

We have discussed Overlapping Subproblems and Optimal Substructure properties in Set 1 and Set 2 respectively. We also discussed one example problem in Set 3. Let us discuss Longest Common Subsequence (LCS) problem as one more example problem that can be solved using Dynamic Programming.

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))
```

Coding Question-4

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++ 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());
}
}
```

Coding Question-5

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(limit³) 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 = m² - n²
b = 2 * m * n
 c = m² + n²
because,
 a² = m⁴ + n⁴ – 2 * m² * n²
b² = 4 * m² * n²
c² = m⁴ + n⁴ + 2* m² * n²
```

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)
```

Coding Question-6

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))
```

Coding Question-7

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.

Also, you can study other Command Line Programming Questions here on our TCS Dashboard.

It is highly advisable to go through Command Line Arguments Post before even looking at the code. Please study this for TCS and come back to this post later.

lease also write the code in C/++/Java/Python in the comments section below
Please comment your own version of code in the comment section below –

```
[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]
```

Coding Question-8

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();
}
```

Coding Question-9

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();
}
```

Coding Question-10

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(); }
```

Coding Question-11

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

It is highly advisable to go through Command Line Arguments Post before even looking at the code. Please study this for TCS and come back to this post later

Please comment your own version of code in the comment section below –

```
#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;

}
```

Coding Question-12

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

Please comment your own version of code in the comment section below –

```
#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;
}
}
```

Coding Question-13

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;
}
```

Coding Question-14

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);
	}
}
```

Coding Question-15

Second Largest Number in an Array using Command Line Programming

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

int main()
{
int arr[50], i, Size;
int first, second;

printf("\n Please Enter the Number of elements in an array : ");
scanf("%d", &Size);

printf("\n Please Enter %d elements of an Array \n", Size);
for (i = 0; i < Size; i++)
{
scanf("%d", &arr[i]);
}

first = second = INT_MIN;
```

Coding Question-16

Sum of odd number in given range using Command Line Programming

Please comment your own version of code in the comment section below –

```
[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;
}
```

Coding Question-17

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

```
#include<stdlib.h>
```

```
#include<stdio.h>

int main(int argc, char *argv[])
{
long n3,i,rem,base=1,decimal,binary=0;

int n1,n2;

n1=atoi(argv[1]);
n2=atoi(argv[2]);

n3=n1+n2;
printf(" The sum of the two numbers are %d",n3);
decimal=n3;
while(n3>0)
{
rem=n3%2;
binary=binary+rem*base;
base=base*10;
n3=n3/2;
}

printf("\n The equivalent binary number is %d",binary);
return 0;
}
```

Coding Question-18

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); }
```

Coding Question-19

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;

}
```

Coding Question-20

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.

Coding Question-21

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"

Solution 1 (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);
```

```
}
```

Coding Question-22

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

Method 1 (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");

}
```

Method 2 (Consecutive Divide by 2 – The traditional way in which we convert a number to Binary)

Coding Question-23

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)

This is a practice question comment down the answer below

```
#include<stdio.h>
#include<stdlib.h>
int main(int c,char *v[])
{
int n1=atoi(v[1]),n2=8,res=0;
while(n2!=0)
{
res=res+n1;
n2--;
}
printf("%d",res);
}
```

Coding Question-24

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.

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
```

```
int n1,n2=31,res=0;

n1=atoi(argv[1]);

while(n2!=0)
{
res=res+n1;
n2--;
}

printf("%d",res);
}
```

Coding Question-25

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);
```

Coding Question-26

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;
}
```

Coding Question-27

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

```
#include < stdio.h >
#include < conio.h >
void main(int argc,char *argv[])
{
int len;
char *ptr;
clrscr();
if ( argc != 2)
{
printf("\n Invalid arguments ");
exit(0);
}
else
{
ptr=argv[1];
len=0;
while ( *(ptr+len) != '\0')
{
len++;
}
printf("\n Length is %d",len);
}
}
```

Coding Question-28

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

Please also comment the code in other languages below –

```
#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;

}

}
```

Coding Question-29

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) );
}
}
```

Coding Question-30

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));
}
}

// This code is contributed by Saket Kumar
```

Coding Question-31

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)

Coding Question-32

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;

}
```

Coding Question-33

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;

}
```

Coding Question-34

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); }
```

Coding Question-35

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

Coding Question-36

Write a Program to remove vowels from a string?

Please comment down your answer.

Coding Question-37

Write a Program for Matrix Multiplication(Universal Program)

Please comment down the code, we will add here.