Coding

1. Program to Reverse a Number in C.

```
#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;
}
```

2. GCD of two numbers

```
#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;
}
```

3. **Check whether a number can be expressed as a sum of two prime numbers**

For example, the number 34 is given as input.

34 = 3 + 31
34 = 5 + 29
34 = 11 + 23
34 = 17 + 17

```
#include <stdio.h>
int sum_of_two_primes(int n);

int main()
{
int n, i;
printf("Enter the number: ");
scanf("%d", &n);
int flag = 0;
for(i = 2; i <= n/2; ++i)
{
// Condition for i to be prime
if (sum_of_two_primes(i) == 1)
{
if (sum_of_two_primes(n-i) == 1)
{
```

```
printf("\n%d can be expressed as the sum of %d and %d\n\n", n, i, n - i);
flag = 1;
}
}
}

if (flag == 0)
printf("%d cannot be expressed as the sum of two prime numbers\n", n);

return 0;
}

//function to check if a number is prime or not
int sum_of_two_primes(int n)
{
int i, isPrime = 1;
for(i = 2; i <= n/2; ++i)
{
if(n % i == 0)
{
isPrime = 0;
break;
}
}
return isPrime;
}
```

4. **Remove brackets from an algebraic string/expression**

Code -
Test case:
Input:x-(p+q)+(y-a)
Output:x-p+q+y-a

```
#include
int main()
```

```
{
int i=0,c=0,j=0;
char a[100],b[100];

printf("\nEnter the string : ");
scanf("%s",a);
while(a[i]!='\0')
{
if((a[i]=='(') && (a[i-1]=='-'))
{
(c==0)?j=i:j=c;
while(a[i]!=')')
{
if(a[i+1]=='+')
b[j++]='-';
else if(a[i+1]=='-')
b[j++]='+';
else if(a[i+1]!=')')
b[j++]=a[i+1];
i++;
}
c=j+1;
}
else if(a[i]=='(' && a[i-1]=='+')
{
(c==0)?j=i:j=c;
while(a[i]!=')')
{
b[j++]=a[i+1];
i++;
}
j–;
c=j+1;
}
else if(a[i]==')')
{
i++;
continue;
}
else
{
b[j++]=a[i];
}
i++;
}
b[j]='\0′;
printf("%s",b);
return 0;
}
```

5. **Finding all the roots of a quadratic equation**

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

int main()
{
double a, b, c, discriminant, root1, root2, realPart, imaginaryPart;

printf("Enter coefficients a, b and c: ");
scanf("%lf %lf %lf",&a, &b, &c);

discriminant = b*b-4*a*c;

// condition for real and different roots
if (discriminant > 0)
{
// sqrt() function returns square root
root1 = (-b+sqrt(discriminant))/(2*a);
root2 = (-b-sqrt(discriminant))/(2*a);

printf("root1 = %.2lf and root2 = %.2lf",root1 , root2);
}
//condition for real and equal roots
else if (discriminant == 0)
{
root1 = root2 = -b/(2*a);

printf("root1 = root2 = %.2lf;", root1);
}

// if roots are not real
else
{
realPart = -b/(2*a);
imaginaryPart = sqrt(-discriminant)/(2*a);
printf("root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi", realPart, imaginaryPart,
realPart, imaginaryPart);
}

return 0;
}
```

6. **SAMPLE PROGRAM TO PRINT ALL INTEGERS USING COMMAND LINE ARGUMENTS**

```
// Program to print all value of command-line argument once we get the value
from command line we can use them to solve our problem

#include <stdio.h>
int main(int argc, char *argv[])
{
int a,b;
int i;
if(argc<2) {
printf("please use \"prg_name value1 value2 ... \"\n");
return -1;
}
for(i=1; i<argc; i++) {
printf("arg[%2d]: %d\n",i,atoi(argv[i]));
}
return 0;
}
```

7. **Find all possible permutations in which 'n' people can occupy 'r' seats in a theater**

For example,
Input:
Number of people: 5
Number of Rows: 3
Output:
The total number of ways in which 'n' people can be seated in 'r' seats = 60.

Calculation:
P(n,r) =P(5,3)
=5! /(5?3)! = 5! / (5 ? 3)!
= 120 / 2 = 60

```
// C program to find all possible permutations in which n people can occupy r
seats in a theater

#include<stdio.h>

// Function to find the factorial of the number
int fact(long int x)
{
        long int f=1,i;
        for(i=1;i<=x;i++)
                {
                        f=f*i;
                }
        return f;
}

int main()
{
        long int n,r,p,temp;
        long int num,den;
        // Enter the number of seats
        printf("Enter the number of seats available : ");
        scanf("%ld",&r);
        // Enter the number of people
        printf("nEnter the number of persons : ");
        scanf("%ld",&n);
        // Base condition
        // Swap n and r
        if(n < r)
        {
                temp=n;
                n=r;
                r=temp;
        }
        num=fact(n);
        den=fact(n-r);
        p=num/den;
        printf("nNumber of ways people can be seated : ");
        printf("%ld",p);
}
```

8. **Count Number of Digits in an Integer**

```
#include <stdio.h>
int main()
{
int n;
int count = 0;
printf("\nEnter the number: ");
scanf("%d", &n);
while(n != 0)
{
n = n/10;
++count;
}
printf("\nNumber of digits: %d\n", count);
}
```

9. **Finding factors of a number in C**

```
#include <stdio.h>

int main()
{
int num;
printf("\nEnter the number : ");
scanf("%d",&num);
int i,count = 0;
printf("\nThe factors of %d are : ",num);
for(i = 1;i <= num; i++)
{
if(num % i == 0)
{
++count;
printf("%d ",i);
}
}
printf("\n\nTotal factors of %d : %d\n",num,count);
}
```

10. **Find the number of times digit 3 occurs in each and every number from 0 to n**

For example,
Input: 100
Output: 20

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

```
#include <stdio.h>

int count_3s(int n)
{
        int count = 0;
        while (n > 0)
        {
                if (n % 10 == 3)
                {
                        count++;
                }
     n = n / 10;
        }
return count;
}

int count_in_range(int n)
{
        int count = 0 ;
        for (int i = 2; i <= n; i++)
        {
                count += count_3s(i);
        }
        return count;
}
```

```
int main()
{
        int n;
        printf("\nEnter the end value : ");
        scanf("%d", &n);
        printf("\nTotal occurrences of 3 from 0 to %d is %d\n",
n,count_in_range(n));
        return 0;
}
```

11. Program to find the number of integers with exactly 9 divisors

Test cases:
Input:
100

Output:
2
36 100

Divisors of 36 = 1, 2, 3, 4, 6, 9, 12, 18, 36
Divisors of 100 = 1, 2, 4, 5, 10, 20, 25, 50, 100

```
#include
int count_no_of_divisors(int num)
{
int count = 0;
for (int i = 1; i <= num; i++)
{
if (num % i == 0)
count = count + 1;
}
return count;
}
```

```
void check_9_factors(int n)
{
int c = 0;
for (int i = 1; i <= n; i++)
{
if (count_no_of_divisors(i) == 9)
{
printf("%d ", i); c = c + 1;
}
}
printf("\n\nTotal = %d\n", c);
}

int main()
{
int n;
printf("\nEnter the number : ");
scanf("%d", &n);
printf("\nThe number which has exactly 9 divisors : ");
check_9_factors(n);
return 0;
}
```

12. **Diamond pattern printing using numbers**

Input:

```
3 4
```

Output:

```
3
44
555
6666
555
44
3
```

```
#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;

}
```

13. **Remove vowels from a string and return the string with consonants**

```
#include <stdio.h>
int check_vowel(char);
int main()
{
char s[100], t[100];
int c, d = 0;
gets(s);
for(c = 0; s[c] != ‘\0’; c++)
{
if(check_vowel(s[c]) == 0)
{
t[d] = s[c];
d++;
}
}
t[d] = ‘\0’;
strcpy(s, t);
printf(“%s\n”, s);
return 0;
}
int check_vowel(char ch)
{
if (ch == ‘a’ || ch == ‘A’ || ch == ‘e’ || ch == ‘E’ || ch == ‘i’ || ch == ‘I’ || ch ==’o’ ||
ch==’O’ || ch == ‘u’ || ch == ‘U’)
return 1;
else
return 0;
}
```

14. **Find the first non-repeating character in a string**

Sample Input 1:
teeterson
Sample Output 1:
r

```
#include<stdlib.h>
#include<stdio.h>
#define NO_OF_CHARS 256
```

```
int *get_char_count(char *str)
{
        int *count = (int *)calloc(sizeof(int), NO_OF_CHARS);
        int i;
        for (i = 0; *(str+i); i++)
                count[*(str+i)]++;
        return count;
}

int first_non_repeating_character(char *str)
{
        int *count = get_char_count(str);
        int index = -1, i;

for (i = 0; *(str+i); i++)
        {
                if (count[*(str+i)] == 1)
                        {
                                index = i;
                                break;
                        }
        }

        free(count);
        return index;
}

int main()
{
        char str[NO_OF_CHARS];
        printf("\nEnter the string : ");
        scanf("%s",&str);
        int index = first_non_repeating_character(str);
        if (index == -1)
                printf("All the characters are repetitive");
        else
                printf("First non-repeating character is %c", str[index]);
        getchar();
        return 0;
}
```

15. Check if Two Strings are anagrams or not

```
#include <stdio.h>

int check_anagram(char [], char []);

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

printf("Enter two strings : \n");
gets(a);
gets(b);

if (check_anagram(a, b) == 1)
printf("The strings are anagrams\n");
else
printf("The strings are not anagrams\n");

return 0;
}

int check_anagram(char a[], char b[])
{
int first[26] = {0}, second[26] = {0}, c=0;

// Calculating frequency of characters of first string

while (a[c] != '\0')
{
first[a[c]-'a']++;
c++;
}

c = 0;

while (b[c] != ‘\0’)
{
second[b[c]-‘a’]++;
c++;
}

// Comparing frequency of characters

for (c = 0; c < 26; c++)
{
if (first[c] != second[c])
return 0;
}

return 1;
}
```

16. Program to reverse an array

```
#include<stdio.h>
int main()
{
//fill the code;
int n;
scanf(“%d”,&n);
int arr[n];
int i;
for(i = 0; i < n; i++)
{
scanf(“%d”,&arr[i]);
}
printf(“Reversed array is:\n”);
for(i = n-1; i >= 0; i–)
{
printf(“%d\n”,arr[i]);
}
return 0;
}
```

17. Program to print the sum of boundary elements of a matrix

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

int main()
{
        int m, n, sum = 0;
        printf(“\nEnter the order of the matrix : “);
        scanf(“%d %d”,&m,&n);
        int i, j;
        int mat[m][n];
        printf(“\nInput the matrix elements\n”);
        for(i = 0; i < m; i++)
        {
                for(j = 0; j < n; j++)
                        scanf(“%d”,&mat[i][j]);
        }

        printf(“\nBoundary Matrix\n”);
        for(i = 0; i < m; i++)
        {
                for(j = 0; j < n; j++)
                        {
                                if (i == 0 || j == 0 || i == n – 1 || j == n – 1)
                                        {
                                                printf(“%d “, mat[i][j]);
                                                sum = sum + mat[i][j];
                                        }
                        else
                                printf(” “);
                        }
                printf(“\n”);
        }
        printf(“\nSum of boundary is %d”, sum);
}
```

18. **Program to find all the patterns of 0(1+)0 in the given string**

0(1+)0 - There should be at least one '1' between the two 0's.
For example, consider the following string.

Input: 01101111010
Output: 3
Explanation:
01101111010 - count = 1
011**011110**10 - count = 2
01101111**010**- count = 3

```
#include <stdio.h>
#include <stdlib.h>
/* Function to count the patterns */
int find_pattern(char str[])
{
char last = str[0];
int i = 1, counter = 0;
while (i < strlen(str))
{
/* We found 1 and last character was ‘0’, state change*/
if (str[i] == ‘1’ && last == ‘0’)
{
```

```
while (str[i] == ‘1’)
i++;
/* After the stream of 1’s, we got a ‘0’, counter incremented*/
if (str[i] == ‘0’)
counter++;
}
/* Store the last character */
last = str[i];
i++;
}
return counter;
}

int main()
{
        char str[50];
        printf(“nEnter the string : “);
        gets(str);
        printf(“nNumber of patterns found : %d”, find_pattern(str));
        printf(“n”);
        return 0;
}
```

19. **Program to count the number of even and odd elements in an array**

```
#include<stdio.h>
int main()
{
//fill your code
int n;
scanf(“%d”,&n);
int arr[n];
for(int i = 0; i < n; i++)
{
```

```
scanf(“%d”,&arr[i]);
}
int count_odd =0, count_even = 0;
for(int i = 0; i < n; i++)
{
if(arr[i] % 2 == 1)
count_odd++;
else
count_even++;
}
printf(“Odd: %d”,count_odd);
printf(“\nEven: %d”,count_even);
return 0;
}
```

20. **Program to sort a string in alphabetical order**

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

int main ()
{
        char string[100];
  printf("\n\t Enter the string : ");
        scanf(“%s”,string);
        char temp;
        int i, j;
        int n = strlen(string);
        for (i = 0; i < n-1; i++) {
                for (j = i+1; j < n; j++) {
                        if (string[i] > string[j]) {
                                        temp = string[i];
                                        string[i] = string[j];
                                        string[j] = temp;
                        }
                }
        }

        printf(“The sorted string is : %s”, string);
        return 0;
}
```

21. Array Rotation | Program for Left Rotation of an Array

```
#include <bits/stdc++.h>
using namespace std;
void left_rotate_by_one(int arr[], int n)
{

/* Shift operation to the left */
int temp = arr[0], i;
for (i = 0; i < n - 1; i++)
arr[i] = arr[i + 1];
arr[i] = temp;
}

void array_left_rotate(int arr[], int no_of_rotations, int n)
{
for (int i = 0; i < no_of_rotations; i++)
left_rotate_by_one(arr, n);   // Function is called for no_of_rotations times
}

int main()
{
int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
int n = sizeof(arr) / sizeof(arr[0]);   // Finding the size of the array
cout<<"\nArray elements before rotating : \n";
for(int i = 0; i < n; i++)
{
cout<<arr[i]<<"\t";    // Printing the array elements
}
int no_of_rotations = 1;  // Number of rotations to take place
array_left_rotate(arr, no_of_rotations, n);
cout<<"\n\nArray elements after rotating : \n";
for(int i = 0; i < n; i++)
{
cout<<arr[i]<<"\t";   // Printing the array elements after rotation of elements
}
cout<<"\n";
return 0;
}
```

22. **Array Rotation | Program for Right Rotation of an Array**

```
#include <bits/stdc++.h>
using namespace std;
void right_rotate_by_one(int arr[], int n)
{

/* Shift operation to the right */
int temp = arr[n - 1], i;
for (i = n - 1; i > 0; i--)
arr[i] = arr[i - 1];
arr[0] = temp;
}
void array_right_rotate(int arr[], int no_of_rotations, int n)
{
for (int i = 0; i < no_of_rotations; i++)
right_rotate_by_one(arr, n);     // Function is called for no_of_rotations times
}
int main()
{
int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int n = sizeof(arr) / sizeof(arr[0]);    // Finding the size of the array
cout<<"\nArray elements before rotating : \n";
for(int i = 0; i < n; i++)
{
cout<<arr[i]<<"\t";     // Printing the array elements
}
int no_of_rotations = 2;
array_right_rotate(arr, no_of_rotations, n);
cout<<"\n\nArray elements after rotating : \n";
for(int i = 0; i < n; i++)
{
cout<<arr[i]<<"\t";    // Printing the array elements after rotation of elements
}
cout<<"\n";
return 0;
}
```

23. **Program to find if the given matrix is upper triangular or not**

```
#include <stdio.h>

int main()
{
        int n;
        scanf(“%d”,&n);
        int flag = 0;
        int mat[n][n];
        int i, j;
        for(i = 0; i < n; i++)
                {
                        for(j = 0; j < n; j++)
                        scanf(“%d”,&mat[i][j]);
                }

        for (i = 1; i < n; i++)
                for (j = 0; j < i; j++)
                        if (mat[i][j] != 0)
                                flag = 0;
                        else
                                flag = 1;
        if (flag == 1)
                printf(“Upper Triangular Matrix”);
        else
                printf(“Not an Upper Triangular Matrix”);
return 0;
}
```

24. Program to find if the given matrix is lower triangular or not

```
#include<stdio.h>
#define N 3

int check_lower_triangular_matrix(int mat[N][N])
{
        int i, j;
        for (i = 0; i < N; i++)
                for (j = i + 1; j < N; j++)
                        if (mat[i][j] != 0)
                                return 0;
return 1;
}

int main()
{
        int mat[N][N];
        int i, j;
        for(i = 0; i < N; i++)
                {
                        for(j = 0; j < N; j++)
                                scanf(“%d”,&mat[i]);
                }
        if (check_lower_triangular_matrix(mat))
                printf(“Lower Triangular Matrix”);
        else
                printf(“Not a Lower Triangular Matrix”);
        return 0;
}
```

25. Program to find Largest and Smallest Element in an Array

```
#include<stdio.h>

int main()
{
   printf("\n\n\t\tStudytonight - Best place to learn\n\n\n");
   int a[50], size, i, big, small;

   printf("\nEnter the size of the array: ");
   scanf("%d", &size);

   printf("\n\nEnter the %d elements of the array: \n\n", size);
   for(i = 0; i < size; i++)
   scanf("%d", &a[i]);

   big = a[0]; // initializing
   /*
      from 2nd element to the last element
      find the bigger element than big and
      update the value of big
   */
   for(i = 1; i < size; i++)
   {
      if(big < a[i])   // if larger value is encountered
      {
         big = a[i]; // update the value of big
      }
   }
   printf("\n\nThe largest element is: %d", big);

   small = a[0];   // initializing
```

```
    /*
        from 2nd element to the last element
        find the smaller element than small and
        update the value of small
    */
    for(i = 1; i < size; i++)
    {
        if(small>a[i])   // if smaller value is encountered
        {
            small = a[i];   // update the value of small
        }
    }
    printf("\n\nThe smallest element is: %d", small);
    printf("\n\n\t\t\tCoding is Fun !\n\n\n");
    return 0;
}
```