1) Aim:- program to print fibonacci series using recursion.

Program:-

```
#include <stdio.h>
int fibonacci (int n) {
    if (n <= 1)
        return n;
    return fibonacci (n-1) + fibonacci (n-2);
}
void Printfibonacciseries (int count) {
    for (int i=0; i< count; i++){
        printf ("%d", fibonacci (i));
    }
}
int main () {
    int count;
    printf (" Enter the number of terms: ");
    scanf ("%d", &count);
    printf (" fibonacci series: ");
    print Fibonacciseries (count);
    return 0;
}
```

Algorithm:-

1. start
2. Define a recursive function fibonacci (n) that takes an integer 'n' as input and returns the nth fibonacci number
3. The base cases for the recursion are:
   * if 'n' is 0, return 0.
   * if 'n' is 1, return 1.
4. otherwise, recursively call 'faboracci (n-1)' and 'fibonacci(n-
5. iterate from 0 to 'num_terms-1' and print the Fibonaci nuber for index

2) write a program to check Amstrong or not using recursive function.

Aim:- Program to check the given number is Amstrong or not using recursive.

Program:-

```
#include <stdio.h>
#include <math.h>
int power (int base, int exponent) {
    if (exponent == 0)
        return 1;
    else
        return base* power (base, exponent - 1);
}
int isArmstrong (int num, int originalNum, int numDigits){
    if (num == 0)
        return originalNum == 0;
    int digit = num % 10;
    int sum = power ( digit, numDigits) + isArmstrong (num/10,
                originalDigits)
    return sum == originalNum;
}
int main () {
    int number;
    printf ("Enter a positive integer: ");
    scanf ("%d", &number);
    int numDigits = log10(number) + 1;
    if (isArmstrong (number, number, numDigits))
        printf ("%d is an Armstrong number\n", number);
    else
        printf ("%d is not an Armstrong number", number);
    return 0;
}
```

Algorithm:-

1. start
2. define a recursive function that takes the two argument the number to be checked and the number of digit in that number.
3. The base case for the recursion is when 'number' is 0 in this case, return 0.
4. Add the result to the return value of the recursive call with 'number' divided by 10 and 'numDigits'
5. in the main program, take the input number from the user.
6. calculate the number of digits in the input number.
7. call the recursive function 'is Armstrong' with the input number and the number of digits.
8. Compare the result with the original input number. if they are equal, the number is an Armstrong number, otherwise, it is not.

input:- x=153

output:- 153 is an Armstrong number.

8. write a Program to find the GCD of two numbers using orecursive factorization.

Aim :- Program to find the GCD of two numbers. using orecursive factorization.

Programm :-

```
#include <stdio.h>
int gcd_orecursive (int a, int b) {
    if (b==0)
        oreturn a;
    else
        oreturn gcd_orecursive (b, a%.b);
}
int main() {
    int num1, num2;
    point ("Enter two positive integers: ");
    scanf ("%d %d", &num1, num2);
    if (num1<=0 || num2<=0) {
    point ("Both numbers must be positive integers\n");
    oreturn 0;
}
```

Algorithm :- 1. stcart.

2. Define a orecursive function 'gcd (a,b)' that takes two integers 'a' and b' as input and oreturn their GCD

3. if `b' is equal to 0, oreturn 'a' as the GCD.

4. otherwise, orecursively call 'gcd (b, a%.b)' where a%.b is the oremainder when 'a' divisible by b

5. The base case foor the orecursion is when 'b' becomes 0, in which case, oreturn 'a'

6. stop.

4 write a program to get the largest element on array.

```
#include <stdio.h>
int findLargest (int arr[], int n){
    if (n==0){
        printf ("Array is empty. \n");
        return -1;
    }
  int largest = arr[0];
for (int i=1; i<n; i++) {
    if (arr[i] > largest){
      largest = arr[i];
    }
  }
return largest;
}
int main () {
    int array [] = {10, 5, 7, 15, 3, 8, 20};
    int size = sizeof (array) / sizeof (array [0]);
    int largest_element = findLargest (array, size);
    if largest_element = find largest (array, size);
    if (largest_element != -1){
        print ("The largest element in the array, Largest el
    }
```

Algorithm:-

1. start
2. Begin with defining a function 'findLonget' that takes an integer array 'arr' and its size 'n' as input.
3. check if the array is emtpty if 'n' is 0, print an error message and return a value indicating an error condtion (for example, -1).
4. iterate through the array from the second element (arr[1]) to the last element (arr[n-1]).
5. After iterating through the entire array, longet will contain the longest element.
6. Return 'longest'
7. stop.

Input:- arr[5] = {16, 14, 29, 17, 6}

output:- longest element is 29

5)

input:- n=5

output:- factorial of 5 is 120

5. write a program to find the factorial of a number using recursion in C.

Aim:- Program to find the factorial of a number using recursion

Program:-

```
#include <stdio.h>
unsigned long long factorial (int n) {
    if (n==0)
        return 1;
    else
        return n* factorial (n-1);
}
int main() {
    int num;
    unsigned long long fact;
    printf ("Enter a non-negative integer: ");
    scanf ("%d", &num);
    if (num<0) {
        printf ("Error");
        return 1;
    }
    fact = factorial (num);
    printf ("factorial of %d = %llu\n", num, fact);
    return 0;
```

Algorithm:- 1. start
2. void null function with argument (int num)
3. if num=0 return 1 else return form (num) to (n-1)
4. now keep int num()
5. Assing value 5
6. Print the factorial
7. stop

Input :- str = "ABCD"

output :- str = "ABCD"

6. write a program for to copy one string to another using Reccursion

Aim:- To Print a Program to copy one string to another string using Recursion.

Algorithm:-

1. start
2. use void function. copystring Assign two files source, destination to copy the string from source to destination.
3. Destination = source use the copy string ( )
4. int main() use and print the source and string
5. stop

code:-

```
# include <stdio.h>
Void copystring ( char* source, char* destination)
    {if (*source = '\0') {
        *destination = '\0';
        return 0;
    }
    *destination = *source;
    copystring ( source+1, destination+1);
    }
int main()
    char source [] = " Hello"
    char destination [50];
    copystring (source, destination);
    printf ( " %s " , source)
    printf (" %s" , destination)
    return 0;
    }
```

7 write a Program to Reverse the storing using Reccursion

Aim:- To Print the Reverse of a storing using Reccursion Method.

Algorithm:-

1. start
2. Take the void (NULL) function Name Reverse insped stor, int, int
3. if start >= end oreturn else.
4. chor temp = stor [start]; stor [start] = stor [end]; stor[en
5. Take the int main (){
   and call the function Reverse (chor; int, int)
6. Print the stor.
7. oreturn the function.
8. stop.

CODE:-

```
# include <stdio.h>
# include <string.h>
void oreverse (chor* stor, int start, int end){
  if (start >= end) {
    oreturn;
  }
chor temp = stor [start];
    stor [start] = stor [end];
    stor [end] = temp;
oreverse (stor, start+1, end-1); }
int main (){
  chor stor [] = "hellow world";
  oreverse (stor, 0, strlen(stor)-1);
  print ("%s", stor);
    oreturn 0;
  }
```

Input :- str = "ABCD"

output :- str = "DCBA"

Input :- n = 10

output :- 2, 3, 5, 7.

```
int num = 20;
if (find_Prime(num))
{
printf ("%d is a Prime number", num)
}
else
{
printf ("%d is not a Prime number", num)
}
return 0;
}
```

9)

Input :- n = 17

output :-

17 is prime number

```
int main()
{
  int num=20;
  if (is_Prime(num))
  {
   printf("%d is a Prime number\n", num);
  }
  else
  {
   printf("%d is not a Prime number\n", num);
  }
}
return 0;
}
```

10)

Input :- "malayalam"

output :- its a pallindrome.

10. write a Program for to check it string is pallindrome or not using Reccursion.

Aim:- a Program to check given string is pallindrome or not.

Algorithm:-

1. start.
2. if the string has one or zero characters, it is pallindrome
3. if the first and last character of the string unequal its not a pallindrome.
4. Recursily. check if the string that excludes the first and last character is pallindrome.
5. stop.

Program:-

```
#include <stdio.h>
    bool func(char * str, int start, int end){
        if (start >= end){
            return true;
        }
        if (str[start] != str[end]){
            return false; }
        return func(str, start+1, end-1); }
int main(){
    char str[100];
    scanf("%s", str);
    int length = strlen(str);
    if(func(str, 0, length-1)){
        printf("%s", str);
```

```
}else{
printf("%s is not a pallindrome", str
}
return 0;
}.
```