1. Write a C program to generate Fibonacci Series

```
#include <stdio.h>
#include <conio.h>  // For Turbo C++ to use getch()

void main() {
   int n, first = 0, second = 1, next, i;

   clrscr();  // Clears the screen (specific to Turbo C++)

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

   printf("Fibonacci sequence:\n");

   for (i = 0; i < n; i++) {
      if (i <= 1)
         next = i;
      else {
         next = first + second;
         first = second;
         second = next;
      }
      printf("%d ", next);
   }

   getch();  // Pauses the screen to view the output
}
```

2. Write a C program to determine if the given string is a palindrome or not.

```
#include <stdio.h>
#include <conio.h>   // For Turbo C++ (getch and clrscr)
#include <string.h>  // For string functions

void main() {
   char str[100], rev[100];
   int i, j, len;

   clrscr();  // Clear the screen (specific to Turbo C++)

   printf("Enter a string: ");
   gets(str);  // Read input string from user

   len = strlen(str);  // Calculate length of the string

   // Reverse the string
   for (i = 0, j = len - 1; i < len; i++, j--) {
      rev[i] = str[j];
   }
   rev[i] = '\0';  // Null-terminate the reversed string

   // Compare original string with reversed string
   if (strcmp(str, rev) == 0)
      printf("The string is a palindrome.\n");
   else
      printf("The string is not a palindrome.\n");

   getch();  // Pause the output screen to view results
}
```

3.Write C programs that use both recursive and non-recursive functions i) To find the factorial of

1. a given integer. ii) To find the GCD (greatest common divisor) of two given integers.

a.

```
#include <stdio.h>

#include <conio.h>

void main()

{

  int n, a, b;

  clrscr();

  printf("Enter any number\n");

  scanf("%d", &n);

  a = recfactorial(n);

  printf("The factorial of a given number using recursion is %d \n", a);

  b = nonrecfactorial(n);

  printf("The factorial of a given number using nonrecursion is %d ", b);

  getch();

}

int recfactorial(int x)

{

  int f;

  if(x == 0)

  {

   return(1);

  }

  else

  {
   f = x * recfactorial(x - 1);
   return(f);
  }
}
int nonrecfactorial(int x)
{
  int i, f = 1;
  for(i = 1;i <= x; i++)
  {
     f = f * i;
  }
  return(f);
}
```

b.

```
#include <stdio.h>
int recgcd(int x, int y) {
if (y == 0)return x;
return recgcd(y, x % y);
}
int nonrecgcd(int x, int y) {
while (y != 0) {
int temp = y;
y = x % y;
x = temp;
}
return x;
}
int main() {
```

```
int a, b;
printf("Enter two numbers: ");
scanf("%d %d", &a, &b);
printf("GCD using recursion: %d\n", recgcd(a, b));
printf("GCD using iteration: %d\n", nonrecgcd(a, b));
return 0;
}
```

4 Write a C program to find the roots of a quadratic equation.

```
#include<stdio.h>
#include<math.h>
void main()
{
 float ab,c,r1,r2,d;
 clrscr();
 printf ("enter the values of a b c\n");
 scanf ("%f%f%f",&a,&b,&c);
 d= b*b-4*a*c;
 if (d>0)
 {
  r1 = -b+sqrt (d) / (2*a);
  r2 = -b-sqrt (d) / (2*a);
  printf ("the real roots = %f%f,r1,r2");
 }
 else if (d==0){
   r1 = -b/(2*a);
   r2 = -b/(2*a);
   printf ("roots are equal =%f%f",r1,r2);
   }
  else
   printf ("roots are imaginary");
   getch();
}
```

5. .Write a C program that uses functions to perform the following:

   i) Addition of Two Matrices ii) Multiplication of Two Matrices

```
#include <stdio.h>
void addMatrix(int a[3][3], int b[3][3], int r, int c);
void multiplyMatrix(int a[3][3], int b[3][3], int r1, int c1, int r2, int c2);
int main() {
 int a[3][3], b[3][3], r1, c1, r2, c2, i, j;
 // Input for Matrix A
 printf("Enter number of rows and columns for Matrix A: ");
 scanf("%d %d", &r1, &c1);
 // Input for Matrix B
 printf("Enter number of rows and columns for Matrix B: ");
 scanf("%d %d", &r2, &c2);
 // Check if addition is possible
 if (r1 != r2 || c1 != c2) {
 printf("Matrix addition is not possible.\n");
 }
 // Check if multiplication is possible
 else if (c1 != r2) {
 printf("Matrix multiplication is not possible.\n");
 }
 else {
 // Input elements of Matrix A
 printf("Enter elements of Matrix A:\n");
 for (i = 0; i < r1; i++) {
 for (j = 0; j < c1; j++) {
 scanf("%d", &a[i][j]);
 }
 }
 // Input elements of Matrix B
 printf("Enter elements of Matrix B:\n");
 for (i = 0; i < r2; i++) {
 for (j = 0; j < c2; j++) {
 scanf("%d", &b[i][j]);
 }
 }
 // Perform matrix addition
 addMatrix(a, b, r1, c1);

 // Perform matrix multiplication
 multiplyMatrix(a, b, r1, c1, r2, c2);
 }
 return 0;
}
// Function to add two matrices
void addMatrix(int a[3][3], int b[3][3], int r, int c) {
 int sum[3][3], i, j;
 printf("Matrix Addition Result:\n");
 for (i = 0; i < r; i++) {
 for (j = 0; j < c; j++) {
 sum[i][j] = a[i][j] + b[i][j];
 printf("%d ", sum[i][j]);
 }
 printf("\n");
 }
}
// Function to multiply two matrices
void multiplyMatrix(int a[3][3], int b[3][3], int r1, int c1, int r2, int c2) {
 int product[3][3] = {0}, i, j, k;
 printf("Matrix Multiplication Result:\n");
 for (i = 0; i < r1; i++) {
 for (j = 0; j < c2; j++) {
 for (k = 0; k < c1; k++) {
 product[i][j] += a[i][k] * b[k][j];
 }
 printf("%d ", product[i][j]);
 }
 printf("\n");
 }
}
```

6. Write a C program which copies one file to another.

```
#include <stdio.h>
#include <stdlib.h>int main(void) {
FILE *source, *dest;
int ch; // Use int to properly handle EOF
// Open source file for reading in binary mode
source = fopen("source.txt", "rb");
if (source == NULL) {
perror("Error opening source file");
exit(EXIT_FAILURE);
}
// Open destination file for writing in binary mode
dest = fopen("dest.txt", "wb");
if (dest == NULL) {
perror("Error opening destination file");
fclose(source);
exit(EXIT_FAILURE);
}
// Copy file contents character by character
while ((ch = fgetc(source)) != EOF) {
fputc(ch, dest);
}
printf("File copied successfully.\n");
// Close both files
fclose(source);
fclose(dest);
return 0;
}
```

7. .Write C programs that implement the Selection sort method to sort a given array of integers in ascending order.

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

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

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

```
        // Find the minimum element in the remaining array
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Swap the found minimum element with the first element
        if (minIndex != i) {
            temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }
}

void displayArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void main() {
    int arr[100], n, i;

    clrscr();  // Clear screen (Turbo C++ specific function)
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("\nOriginal array: ");
    displayArray(arr, n);

    selectionSort(arr, n);

    printf("\nSorted array in ascending order: ");
    displayArray(arr, n);

    getch();  // Wait for user input before closing (Turbo C++ specific function)
}
```

8. Write C programs that implements the Bubble sort method to sort a given array of integers in ascending order.

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

void bubbleSort(int arr[], int n) {
    int i, j, temp;

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void displayArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void main() {
    int arr[100], n, i;

    clrscr();  // Clear screen (Turbo C++ specific function)

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

    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("\nOriginal array: ");
    displayArray(arr, n);

    bubbleSort(arr, n);

    printf("\nSorted array in ascending order: ");
    displayArray(arr, n);

    getch();  // Wait for user input before closing (Turbo C++ specific function)
}
```

9. .Write C programs that uses non recursive function to search for a key value in a given list of integers using Linear search.

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

int linearSearch(int arr[], int n, int key) {
    int i;
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            return i; // Key found at index i
        }
    }
    return -1; // Key not found
}
```

```
void main() {
    int arr[100], n, i, key, result;

    clrscr();  // Clear screen (Turbo C++ specific function)

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

    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the key value to search: ");
    scanf("%d", &key);

    result = linearSearch(arr, n, key);

    if (result != -1) {
        printf("Key %d found at index %d.\n", key, result);
    } else {
        printf("Key %d not found in the list.\n", key);
    }

    getch();  // Wait for user input before closing (Turbo C++ specific function)
}
```

10. **.** Write C programs that uses non recursive function to search for a key value in a given list of integers using Binary search.

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

int binarySearch(int arr[], int n, int key) {
   int low = 0, high = n - 1, mid;

   while (low <= high) {
      mid = (low + high) / 2;

      if (arr[mid] == key) {
         return mid; // Key found at index mid
      } else if (arr[mid] < key) {
         low = mid + 1; // Search right half
      } else {
         high = mid - 1; // Search left half
      }
   }
   return -1; // Key not found
}

void main() {
   int arr[100], n, i, key, result;

   clrscr();  // Clear screen (Turbo C++ specific function)

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

```
    printf("Enter %d sorted integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);  // Ensure the array is sorted before using binary search
    }

    printf("Enter the key value to search: ");
    scanf("%d", &key);

    result = binarySearch(arr, n, key);

    if (result != -1) {
        printf("Key %d found at index %d.\n", key, result);
    } else {
        printf("Key %d not found in the list.\n", key);
    }

    getch();  // Wait for user input before closing (Turbo C++ specific function)
}
```