Problem 1

Print the following output.

```
*
**
***
****
*****
```

Code:

```c
// include header files
#include <stdio.h>

// main function
void main()
{
    // initialize an integer to specify how many lines of pattern to be printed
    int lineCount = 5;

    // for loop from 1-lineCount
    for (int i = 1; i <= lineCount; i++)
    {
        // another for loop from 1-i
        for (int j = 1; j <= i; j++)
        {
            printf("*"); // print * on the i'th line j i times
        }

        printf("\n"); // go next line
    }
}
```

Output:

```
*
**
***
****
*****
```

Problem 2

Take a String, integer, char, double input from user and print it.

Code:

```c
// include required header
#include <stdio.h>

void main()
{
    // variable declaration
    char string[100], character;
    int integer;
    double doubleNumber;

    // take string input
    printf("Input String: ");
    scanf("%s", &string);

    // take integer input
    printf("Input Integer: ");
    scanf("%d", &integer);

    // take character input
    printf("Input Character: ");
    scanf("%s", &character);

    // take double input
    printf("Input Double: ");
    scanf("%lf", &doubleNumber);

    // output all the inputs
    printf("Inputted String: %s\n", string);
    printf("Inputted Integer: %d\n", integer);
    printf("Inputted Character: %c\n", character);
    printf("Inputted Double: %lf", doubleNumber);
}
```

Output:

```
Input String: taskphase
Input Integer: 5
Input Character: a
Input Double: 3.1428
Inputted String:
Inputted Integer: 5
Inputted Character: a
Inputted Double: 3.142800
```

Problem 3

Print the sequence shown in Q1 but the number of rows should be decided by the user.

Code:

```c
// include required header file
#include <stdio.h>

void main()
{
    // integer variable to store user input for line count
    int lineCount;

    // take the user input
    printf("Enter the number of lines: ");
    scanf("%i", &lineCount);

    // same as Q1 from here
    for (int i = 1; i <= lineCount; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("*");
        }

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

Output:

```
Enter the number of lines: 3
*
**
***
```

Problem 4

Take a String input and count the number of vowels in it.

Code:

```c
// include required header files
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```c
void main()
{
    // declare / initialize required variables - a string to take user input and an integer
    char string[100];
    int count = 0;

    // take user input
    printf("Enter your String: ");
    scanf("%s", &string);

    // for loop that runs from 0-string length
    for (int i = 0; i < strlen(string); i++)
    {
        // get the current iterating character
        char c = tolower(string[i]);

        //check if current character is a vowel
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')
            count++; // if it is, increment the count by 1
    }

    // print the final count after looping through every character
    printf("Number of vovels in given string \"%s\": %d", string, count);
}
```

Output:

```
Enter your String: teststring
Number of vovels in given string "teststring": 2
```

Problem 5

Check the String if it is Palindrome or not.

Code:

```c
// include required header files
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

void main()
{
    // string variable to take user input
```

```
    char string[100];

    // take user input
    printf("Enter your String: ");
    scanf("%s", &string);

    // calculate string length and store it in an int variable
    int len = strlen(string);

    // for loop to iterate through every character in the string
    for (int i = 0; i < len; i++)
    {
        // current character from left
        char c1 = tolower(string[i]);
        // current character from right
        char c2 = tolower(string[len - i - 1]);

        // if left isn't equal to right, its not palindrome
        if (c1 != c2)
        {
            // print result
            printf("Not palindrome");
            // exit program
            exit(0);
        }
    }

    // if the code made it this far, it means the given string is palindrome
    printf("Palindrome");
}
```

Output:

```
(1) Enter your String: racecar
    Palindrome

(2) Enter your String: notapalindrome
    Not palindrome
```

Problem 6

Create a program that uses pointers to swap the values of two variables.

Code:

```c
// include required header file
#include <stdio.h>

// function to swap the value of 2 variables using pointers
void swap(int *a, int *b)
{
    int t = *a; // initialize a variable t to hold the value of a
    *a = *b; // change the value of a to value of b
    *b = t; // change the value of b to the initial value of a, which is stored in t
}

// main function
void main()
{
    // initialize 2 variables for user input
    int a, b;

    // take user input
    printf("Input your two numbers (a b): ");
    scanf("%d %d", &a, &b);

    // swap the 2 variables using the above declared function
    swap(&a, &b);

    // print the swapped result
    printf("After pointer-swap:\na = %d\nb = %d", a, b);
}
```

Output:

```
Input your two numbers (a b): 2 3
After pointer-swap:
a = 3
b = 2
```

Problem 7

Write a program to sort an array of integers using the bubble sort algorithm.

Code:

```c
// include required header file
#include <stdio.h>

// bubble sort function which takes array and array size as parameters
void bubbleSort(int arr[], int n)
```

```
{
    // for loop that goes from 0-(array size - 1)
    for (int i = 0; i < n; i++)
    {
        // declare a temporary variable t to store if an element was swapped in this loop
        int t = 0;

        // another for loop that runs from 0 to n - i - 1
        for (int j = 0; j < n - i - 1; j++)
        {
            // if the first element is greater than second, swap
            if (arr[j] > arr[j + 1])
            {
                t = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = t;

                t = 1; // change swap variable to 1, indicating we swapped
            }
        }

        // if no more swapping is required, break
        if (t == 0) break;
    }
}

// main function
void main()
{
    // array to take user input, n for number of elements in array and i for loop variable
    int arr[100], n, i;

    // take user input for number of elements in array
    printf("Enter the array size: ");
    scanf("%d", &n);

    // take array elements input
    printf("Enter the array elements: ");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // bubble sort the array
    bubbleSort(arr, n);

    // print the sorted array
    printf("The array after sorting: ");
```

```
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}
```

Output:

```
Enter the array size: 5
Enter the array elements: 4 3 2 5 1
The array after sorting: 1 2 3 4 5
```

Problem 8

Implement a binary search algorithm to find an element in a sorted array

Code:

```
// include required header file
#include <stdio.h>

// define binary search function, taking in the array, array size and the element to be sea
int binarySearch(int *arr, int n, int element)
{
    // left, right, middle variables, used to binary search
    int left = 0;
    int right = n - 1;
    int middle;

    // loop until left is lesser than right
    while (left <= right)
    {
        // calculate the middle
        middle = (left + right) / 2.0;

        // if element is found at middle, return middle
        if (arr[middle] == element)
            return middle;

        // if element is lesser than the element at middle index, cut short the left interv
        else if (arr[middle] < element)
            left = middle + 1;

        // this means the element is greater than the element at the middle index, so cut sh
        else
            right = middle - 1;
    }
```

```
    // element not found
    return -1;
}

// main function
void main()
{
    // required variables, taken from user input
    int arr[100], n, i, e;

    // input array size from user
    printf("Enter the array size: ");
    scanf("%d", &n);

    // input array elements from user
    printf("Enter the array elements: ");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // input search element from user
    printf("Enter the search element: ");
    scanf("%d", &e);

    // do the binary search and store the returned index in the pos variable
    int pos = binarySearch(arr, n, e);

    // if pos is -1, that means the element wasn't found in the array. Otherwise it means it
    if (pos < 0)
        printf("Element \"%d\" was not found in the array.", e);
    else
        printf("Element \"%d\" was found in the array at position %d.", e, pos);
}
```

Output:

```
(1) Enter the array size: 5
    Enter the array elements: 1 2 3 4 5
    Enter the search element: 2
    Element "2" was found in the array at position 1.

(2) Enter the array size: 5
    Enter the array elements: 1 2 3 4 5
    Enter the search element: 6
    Element "6" was not found in the array.
```

Problem 9

Take each element of the 4x4 matrix from the user and print it, then add each element from the 2D matrix and print the sum, print the sum of both the diagonals

Code:

```c
// include required header file
#include <stdio.h>

// main function
void main()
{
    // variables to store array, and the sums required to compute
    int arr[4][4], i, j, totalSum = 0, d1Sum = 0, d2Sum = 0, dSum = 0;

    // input array elements from the user
    printf("Enter array elements: ");
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            scanf("%d", &arr[i][j]);

    // for loops to iterate through every element of the array
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
        {
            // get current looping element given its indices i,j
            int current = arr[i][j];

            // add to the total sum
            totalSum += current;

            // primary diagonal
            if (i == j)
                d1Sum += current;

            // secondary diagonal
            else if (j == 3 - i)
                d2Sum += current;
        }
    // sum of elements of both diagonals
    dSum = d1Sum + d2Sum;

    // print the array elements
    printf("Array elements are: ");
    for (i = 0; i < 4; i++)
```

```
        for (j = 0; j < 4; j++)
            printf("%d ", arr[i][j]);

    // print the computed sums
    printf("\n\nTotal sum of the matrix is: %d\n", totalSum);
    printf("Sum of the diagonal 1 elements is: %d\n", d1Sum);
    printf("Sum of the diagonal 2 elements is: %d\n", d2Sum);
    printf("Sum of both the diagonal elements is: %d\n", dSum);
}
```

Output:

```
Enter array elements: 1 2 3 4 5 6 7 8 9 10 11 12 12 14 15 16
Array elements are: 1 2 3 4 5 6 7 8 9 10 11 12 12 14 15 16

Total sum of the matrix is: 135
Sum of the diagonal 1 elements is: 34
Sum of the diagonal 2 elements is: 33
Sum of both the diagonal elements is: 67
```

Problem 10

Write a recursive function to calculate the factorial of a number

Code:

```
// include required header file
#include <stdio.h>

// function to calculate factorial of num
int factorial(int num)
{
    // if the number is greater than or equal to 1, return num times factorial of num - 1, o
    return num >= 1 ? num * factorial(num - 1) : 1;
}

// main function
void main()
{
    // int variable to hold user input
    int num;

    // take user input
    printf("Enter the number: ");
    scanf("%d", &num);
```

```
    // calculate factorial (using recursion)
    int fact = factorial(num);

    // print the result
    printf("Factorial of %d is %d.", num, fact);
}
```

Output:

```
Enter the number: 5
Factorial of 5 is 120.
```

Problem 11

Implement Bubble sort and Selection sort. Take the input from the user.

Code:

```
// include required header files
#include <stdio.h>
#include <stdlib.h>

// bubble sort function which takes array and array size as parameters
void bubbleSort(int arr[], int n)
{
    // for loop that goes from 0-(array size - 1)
    for (int i = 0; i < n; i++)
    {
        // declare a temporary variable t to store if an element was swapped in this loop
        int t = 0;

        // another for loop that runs from 0 to n - i - 1
        for (int j = 0; j < n - i - 1; j++)
        {
            // if the first element is greater than second, swap
            if (arr[j] > arr[j + 1])
            {
                t = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = t;

                t = 1; // change swap variable to 1, indicating we swapped
            }
        }

        // if no more swapping is required, break
```

```
        if (t == 0) break;
    }
}

// selection sort function which takes array, current index and array size as parameters
void selectionSort(int arr[], int i, int n)
{
    // lowest position and highest number variables for sorting purposes
    int lowestPos = i;
    int lowestNum = -1;

    // loop from i to n - 1
    for (int j = i; j < n; j++)
        // if current element is the first element to be iterated or if the element is lesse
        if (lowestNum < 0 || arr[j] < lowestNum)
        {
            // change the lowest number to this and store its position in lowestPos
            lowestPos = j;
            lowestNum = arr[j];
        }

    // if lowestPos is greater than i, swap the two elements at respective indices
    if (lowestPos > i)
    {
        int t;
        t = arr[i];
        arr[i] = arr[lowestPos];
        arr[lowestPos] = t;
    }

    // if i greater than n, sorting is complete
    if (i >= n)
        return;

    // sorting incomplete, so continue sorting
    else
        selectionSort(arr, i + 1, n);
}

// main function
void main()
{
    // variables to store array elements, array length and user choice for sorting algo
    int arr[100], n, i, choice;

    // input array size from user
```

```
    printf("Enter the array size: ");
    scanf("%d", &n);

    // input array elements from user
    printf("Enter the array elements: ");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // input sorting algo choice from user
    printf("Enter 0 for bubble sort or 1 for selection sort: ");
    scanf("%d", &choice);

    // if user selected bubble sort, sort array using bubbleSort() function
    if (choice == 0)
        bubbleSort(arr, n);

    // else if user selected selection sort, sort array using selectionSort() function
    else if (choice == 1)
        selectionSort(arr, 0, n);

    // in case of invalid choice, let the user know and exit program
    else
    {
        printf("Invalid choice. Valid choices are 0 and 1.");
        exit(0);
    }

    // print the sorted array
    printf("The array after sorting: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}
```

Output:

```
(1) Enter the array size: 5
    Enter the array elements: 4 3 5 2 1
    Enter 0 for bubble sort or 1 for selection sort: 0
    The array after sorting: 1 2 3 4 5

(2) Enter the array size: 5
    Enter the array elements: 4 3 5 2 1
    Enter 0 for bubble sort or 1 for selection sort: 1
    The array after sorting: 1 2 3 4 5

(3) Enter the array size: 5
    Enter the array elements: 4 3 5 2 1
```

```
Enter 0 for bubble sort or 1 for selection sort: 2
Invalid choice. Valid choices are 0 and 1.
```

Problem 12

Write a program to read a text file, count the number of words in it, and display the result.

Code:

```c
// include required header file
#include <stdio.h>

// main function
void main()
{
    // create buffer for file read, and variable current for loop lateron
    char file[100], current;

    // variables for keeping track of word count and loop
    int wordCount = 0, i = 1;

    // pointer for reading file
    FILE* ptr;

    // read the file
    ptr = fopen("12-file.txt", "r");

    // write contents of the file to buffer
    fgets(file, 100, ptr);

    // initialize current variable
    current = file[i];

    // loop until the end of the stream
    while (current != '\0')
    {
        // if the current character is a space, one more word is counted
        if (current == ' ')
            wordCount++;

        // change current and i for next iteration
        current = file[i];
        i++;
    }
```

```
    // edge case where if the stream ends with a space, one more than the number is counted
    if (file[i - 2] != ' ')
        wordCount++;

    // print the number of words in the string
    printf("The word count in the file \"12-file.txt\" is %d.", wordCount);
}
```

Output:

```
The word count in the file "12-file.txt" is 7.
```