Q.1 1 D integer array
Write a C program using functions and pointers for the following
1. Read and display n numbers
2. Read and display odd positioned elements
3. Display the even numbers from the set of integers
4. Display maximum number from the set of integers
5. Calculate the sum and average of n numbers

```c
#include <stdio.h>

void readAndDisplay(int *arr, int n);
void displayOddPositioned(int *arr, int n);
void displayEvenNumbers(int *arr, int n);
void displayMaximumNumber(int *arr, int n);
void calculateSumAndAverage(int *arr, int n);

int main()
{
    int n;

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

    int arr[n];

    printf("Enter %d elements:\n", n);
    readAndDisplay(arr, n);

    printf("Odd positioned elements:\n");
    displayOddPositioned(arr, n);

    printf("Even numbers:\n");
    displayEvenNumbers(arr, n);

    printf("Maximum number:\n");
    displayMaximumNumber(arr, n);

    printf("Sum and Average:\n");
    calculateSumAndAverage(arr, n);

    return 0;
}

void readAndDisplay(int *arr, int n)
{
```

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

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

void displayOddPositioned(int *arr, int n)
{
    printf("Odd positioned elements: ");
    for (int i = 1; i < n; i += 2)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void displayEvenNumbers(int *arr, int n)
{
    printf("Even numbers: ");
    for (int i = 0; i < n; i++)
    {
        if (arr[i] % 2 == 0)
        {
            printf("%d ", arr[i]);
        }
    }
    printf("\n");
}

void displayMaximumNumber(int *arr, int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
```

```
        }
    }
    printf("Maximum number: %d\n", max);
}

void calculateSumAndAverage(int *arr, int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
    {
        sum += arr[i];
    }

    float average = (float)sum / n;

    printf("Sum: %d\n", sum);
    printf("Average: %.2f\n", average);
}
```

Output-

Enter the number of elements: 5

Enter 5 elements:

Enter 5 numbers:

1 3 7 11 45

Entered numbers: 1 3 7 11 45

Odd positioned elements:

Odd positioned elements: 3 11

Even numbers:

Even numbers:

Maximum number:

Maximum number: 45

Sum and Average:

Sum: 67

Average: 13.40

Q2. 2 D integer array
Write a C program using functions and pointers for the following
1. Read and display n*n matrix
2. Calculate the row total of a given matrix
3. Check whether the given matrix is identity matrix or not

```c
#include <stdio.h>

void readMatrix(int (*arr)[10], int n);
void displayMatrix(int (*arr)[10], int n);
int calculateRowTotal(int (*arr)[10], int n, int row);
int isIdentityMatrix(int (*arr)[10], int n);

int main()
{
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    int arr[10][10];

    printf("Enter the elements of the matrix:\n");
    readMatrix(arr, n);

    printf("The elements are:\n");
    displayMatrix(arr, n);

    int rowToCalculate = 0;
    int rowTotal = calculateRowTotal(arr, n, rowToCalculate);
    printf("Total of Row %d: %d\n", rowToCalculate + 1, rowTotal);


    if (isIdentityMatrix(arr, n))
    {
        printf("The matrix is an identity matrix.\n");
    }
    else
    {
        printf("The matrix is not an identity matrix.\n");
    }

    return 0;
}

void readMatrix(int (*arr)[10], int n)
{
```

```
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            scanf("%d", (*(arr + i) + j));
        }
    }
}

void displayMatrix(int (*arr)[10], int n)
{
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            printf("%d\t", *(*(arr + i) + j));
        }
        printf("\n");
    }
}

int calculateRowTotal(int (*arr)[10], int n, int row)
{
    int total = 0;
    for (int j = 0; j < n; ++j)
    {
        total += *(*(arr + row) + j);
    }
    return total;
}

int isIdentityMatrix(int (*arr)[10], int n)
{
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            if ((i == j && *(*(arr + i) + j) != 1) || (i != j && *(*(arr + i) +
j) != 0))
            {
                return 0;
            }
        }
    }
    return 1;
```

```
}
```

Output-

Enter the value of n: 3

Enter the elements of the matrix:

4 5 6 7 8 9 10 11 12

The elements are:

4 5 6

7 8 9

10 11 12

Total of Row 1: 15

The matrix is not an identity matrix.

Q3. 1 D Char array
Write a C program using functions and pointers for the following
1. Read and display a string
2. Without using string builtin functions, calculate the string length
3. Without using string builtin functions, reverse the string
4. Without using string builtin functions, copy one string into other
5. Read a string and check whether the given character is present or not. If present, count the number of times, it is repeated

```c
#include <stdio.h>

// Function prototypes
void readAndDisplayString(char *str);
int calculateStringLength(const char *str);
void reverseString(char *str);
void copyString(char *dest, const char *src);
int countCharacter(const char *str, char ch);

int main()
{
    char inputString[100];
    char targetString[100];
    char searchChar;

    printf("Enter a string: ");
    readAndDisplayString(inputString);


    printf("Length of the string: %d\n", calculateStringLength(inputString));


    reverseString(inputString);
    printf("Reversed string: %s\n", inputString);

    copyString(targetString, inputString);
    printf("Copied string: %s\n", targetString);
    printf("Enter a character to search: ");
    scanf(" %c", &searchChar);

    int count = countCharacter(inputString, searchChar);
    printf("Character '%c' is present %d times in the string.\n", searchChar,
count);

    return 0;
}
```

```c
void readAndDisplayString(char *str)
{
    scanf("%s", str);
    printf("Entered string: %s\n", str);
}

int calculateStringLength(const char *str)
{
    int length = 0;
    while (*str != '\0')
    {
        length++;
        str++;
    }
    return length;
}

void reverseString(char *str)
{
    char *start = str;
    char *end = str + calculateStringLength(str) - 1;

    while (start < end)
    {

        char temp = *start;
        *start = *end;
        *end = temp;

        start++;
        end--;
    }
}

void copyString(char *dest, const char *src)
{
    while ((*dest++ = *src++) != '\0')
        ;
}

int countCharacter(const char *str, char ch)
{
    int count = 0;
    while (*str != '\0')
```

```
    {
        if (*str == ch)
        {
            count++;
        }
        str++;
    }
    return count;
}
```

Output-

Enter a string: nayan

Entered string: nayan

Length of the string: 5

Reversed string: nayan

Copied string: nayan

Enter a character to search: a

Character 'a' is present 2 times in the string.

Q4. 2 D Char array
Write a C program using functions and pointers for the following
1. Read and display n names
2. Implement bubble sort for n names

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

// Function prototypes
void read(char (*names)[50], int n);
void display(char (*names)[50], int n);
 void bubbleSortNames(char (*names)[50], int n);

int main()
{
    int n;
    printf("Enter the number of names: ");
    scanf("%d", &n);

    char names[n][50];

    read(names, n);
    printf("the entered names are\n");
    display(names,n);
    bubbleSortNames(names, n);

    printf("\nSorted names:\n");
    display(names, n);

    return 0;
}

void read(char (*names)[50], int n)
{
    printf("Enter %d names:\n", n);
    for (int i = 0; i < n; i++)
    {
        scanf("%s", names[i]);
    }


}
void display(char (*names)[50], int n)
{
    for (int i = 0; i < n; i++)
    {
```

```
        printf("%s\n", names[i]);
    }
}

void bubbleSortNames(char (*names)[50], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (strcmp(names[j], names[j + 1]) > 0)
            {
                // Swap names if they are in the wrong order
                char temp[50];
                strcpy(temp, names[j]);
                strcpy(names[j], names[j + 1]);
                strcpy(names[j + 1], temp);
            }
        }
    }
}
```

Output-

Enter the number of names: 2

Enter 2 names:

nayan

aryan

the entered names are

nayan

aryan

Sorted names:

aryan

nayan