Name :Pravendra Jasawat

Section :Al -2

University roll no.2315001667

Class roll no.46

Course :Btech CSE

C-programming

Weekly Questions

Week -5

Q. 1 Write a program to print the following patterns:

a) *****

*****

*****

*****

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change the value of rows to adjust the size of the square

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
****
****
****
****

--------------------------------
Process exited after 0.1235 seconds with return value 0
Press any key to continue . . .
```

B)

12345

12345

12345

12345

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change the value of rows to adjust the size of the rectangle

    for (int i = 0; i < rows; i++) {
        for (int j = 1; j <= 5; j++) {
            printf("%d", j);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
12345
12345
12345
12345

--------------------------------
Process exited after 0.09099 seconds with return value 0
Press any key to continue . . .
```

c)

1

12

123

1234

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change the value of rows to adjust the size of the triangle

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("%d", j);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
1
12
123
1234

--------------------------------
Process exited after 0.05504 seconds with return value 0
Press any key to continue . . .
```

D)

1

22

333

4444

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change the value of rows to adjust the size of the triangle

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("%d", i);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
1
22
333
4444

--------------------------------
Process exited after 0.1431 seconds with return value 0
Press any key to continue . . .
```

Open a new tab
Alt+Click to split the current window
Shift+Click to open a new window

E)

*

**

***

****

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change this value to adjust the number of rows

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
*
**
***
****

--------------------------------
Process exited after 0.08374 seconds with return value 0
Press any key to continue . . .
```

F)

A

AB

ABC

ABCD

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change this value to adjust the number of rows

    for (int i = 0; i < rows; i++) {
        // Print spaces
        for (int j = 0; j < rows - i - 1; j++) {
            printf(" ");
        }

        // Print characters
        for (int j = 0; j <= i; j++) {
            printf("%c", 'A' + j);
        }

        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
   A
  AB
 ABC
ABCD

--------------------------------
Process exited after 0.06272 seconds with return value 0
Press any key to continue . . .
```

G)

1

23

456

78910

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change this value to adjust the number of rows
    int count = 1;

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("%d", count);
            count++;
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
1
23
456
78910

--------------------------------
Process exited after 0.0521 seconds with return value 0
Press any key to continue . . .
```

H)

1

10

101

1010

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 4; // You can change this value to adjust the number of rows

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            if (j % 2 == 0) {
                printf("0");
            } else {
                printf("1");
            }
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
1
10
101
1010

--------------------------------
Process exited after 0.1077 seconds with return value 0
Press any key to continue . . .
```

I)

5

54

543

5432

54321

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 5; // You can change this value to adjust the number of rows

    for (int i = 0; i < rows; i++)
    {
        for (int j = rows; j >= rows - i; j--)
        {
            printf("%d", j);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
5
54
543
5432
54321

--------------------------------
Process exited after 0.09314 seconds with return value 0
Press any key to continue . . .
```

J)

*****

* *

* *

* *

*****

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 5; // You can change this value to adjust the number of rows

    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= rows; j++) {
            if (i == 1 || i == rows || j == 1 || j == rows)
            {
                printf("*");
            } else {
                printf(" ");
            }
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
*****
*   *
*   *
*   *
*****

--------------------------------
Process exited after 0.1026 seconds with return value 0
Press any key to continue . . .
```

K)

*

**

***

****

*****

Ans.

```c
#include <stdio.h>

int main() {
    int rows = 5;

    for (int i = 1; i <= rows; i++)
    {
            for (int j = 1; j <= rows - i; j++)
        {
            printf(" ");
        }
        for (int k = 1; k <= i; k++)
        {
            printf(" *");
        }
        printf("\n");
    }

    return 0;
}
```

Compile Log | Debug | Find Results | Close

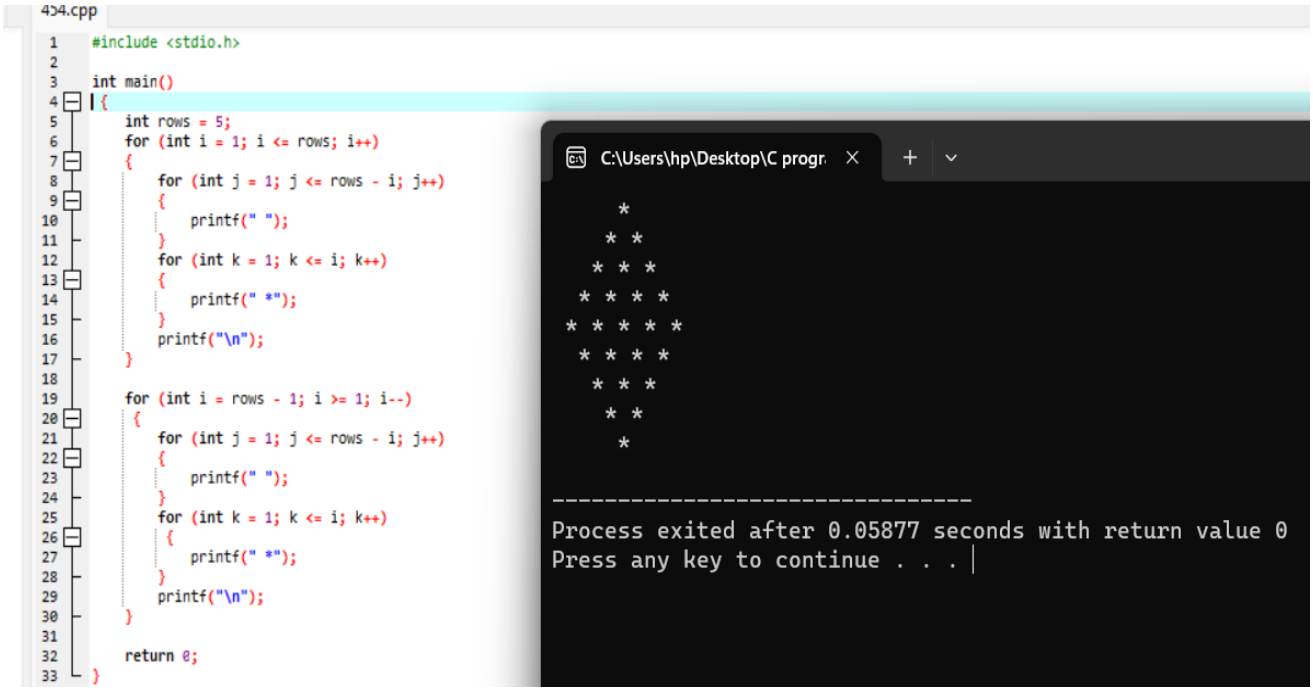
```
C:\Users\hp\Desktop\C progr
    *
   * *
  * * *
 * * * *
* * * * *

--------------------------------
Process exited after 0.06088 seconds with return value 0
Press any key to continue . . .
```

M)

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

Ans.



Week 6

Q. 1 Write a menu driven program to insert and delete elements of kth position to an array of size N.

```c

#include <stdio.h>

// Function to insert element at kth position
void insertElement(int arr[], int *n, int k, int element) {
    if (*n >= k) {
        for (int i = *n; i >= k; i--) {
            arr[i] = arr[i - 1];
        }
        arr[k - 1] = element;
        (*n)++;
        printf("Element %d inserted at position %d.\n", element, k);
    } else {
        printf("Invalid position. Position should be between 1 and %d.\n", *n + 1);
    }
}

// Function to delete element from kth position
void deleteElement(int arr[], int *n, int k) {
    if (*n > 0 && k <= *n) {
        int deletedElement = arr[k - 1];
        for (int i = k - 1; i < *n - 1; i++) {
            arr[i] = arr[i + 1];
        }
        (*n)--;
        printf("Element %d deleted from position %d.\n", deletedElement, k);
    } else {
        printf("Invalid position. Position should be between 1 and %d.\n", *n);
    }
}
```

Compile Log | Debug | Find Results | Close

```c

// Function to display the array
void displayArray(int arr[], int n) {
    printf("Current Array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[100];  // Assuming a maximum array size of 100
    int n = 0;     // Current size of the array

    int choice, k, element;

    do {
        printf("\nMenu:\n");
        printf("1. Insert element at kth position\n");
        printf("2. Delete element from kth position\n");
        printf("3. Display array\n");
        printf("4. Exit\n");

        printf("Enter your choice (1-4): ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the position (1 to %d): ", n + 1);
                scanf("%d", &k);
```

rces | Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
```

```
            printf("Enter the element to insert: ");
            scanf("%d", &element);
            insertElement(arr, &n, k, element);
            break;
        case 2:
            printf("Enter the position (1 to %d): ", n);
            scanf("%d", &k);
            deleteElement(arr, &n, k);
            break;
        case 3:
            displayArray(arr, n);
            break;
        case 4:
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice. Please enter a number betw
    }
} while (choice != 4);

return 0;
}
```

Compile Log | Debug | Find Results | Close

```
C:\Users\hp\Desktop\C progr

Menu:
1. Insert element at kth position
2. Delete element from kth position
3. Display array
4. Exit
Enter your choice (1-4): 1
Enter the position (1 to 1): 2
Enter the element to insert: 10
Invalid position. Position should be between 1 and 1.

Menu:
1. Insert element at kth position
2. Delete element from kth position
3. Display array
4. Exit
Enter your choice (1-4):
```

Q2.Write the program to print the biggest and smallest element in an array

Ans.

[*] 454.cpp

```
#include <stdio.h>

void findMinMax(int arr[], int size, int *min, int *max)
{
    *min = *max = arr[0]; // Assume the first element as both min and max initially

    for (int i = 1; i < size; i++) {
        if (arr[i] < *min) {
            *min = arr[i]; // Update min if current element is smaller
        }
        if (arr[i] > *max) {
            *max = arr[i]; // Update max if current element is larger
        }
    }
}

int main() {
    int size;

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

    int arr[size];
```

```
int main() {
    int size;

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

    int arr[size];

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

    int min, max;
    findMinMax(arr, size, &min, &max);

    printf("Smallest element: %d\n", min);
    printf("Largest element: %d\n", max);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr

Enter the size of the array: 5
Enter 5 elements:
10
20
30
40
50
Smallest element: 10
Largest element: 50

--------------------------------
Process exited after 8.453 seconds with return value 0
Press any key to continue . . .
```

Q. 3 Write the program to print the sum and average of an array.

[*] 454.cpp

```c
#include <stdio.h>

void calculateSumAndAverage(int arr[], int size, int *sum, float *average)
{
    *sum = 0; // Initialize sum to 0

    for (int i = 0; i < size; i++)
    {
        *sum += arr[i]; // Add each element to the sum
    }

    *average = (float)(*sum) / size; // Calculate the average
}

int main()
{
    int size;

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

    int arr[size];

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

    int sum;
    float average;
    calculateSumAndAverage(arr, size, &sum, &average);

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

    return 0;
}
```

Resources | Compile Log | Debug | Find Results | Close

Compilation results...

C:\Users\hp\Desktop\C progr

```
Enter the size of the array: 4
Enter 4 elements:
10
20
30
40
Sum of elements: 100
Average of elements: 25.00

--------------------------------
Process exited after 8.441 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Write the program to sort an array using bubble sort.

```c
#include <stdio.h>

void bubbleSort(int arr[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                // Swap arr[j] and arr[j+1] if they are in the wrong order
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main()
{
    int size;
```

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

    int arr[size];

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

    // Call the bubbleSort function to sort the array
    bubbleSort(arr, size);

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

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the size of the array: 5
Enter 5 elements:
10
40
60
30
50
Sorted array:
10 30 40 50 60
--------------------------------
Process exited after 11.11 seconds with return value 0
Press any key to continue . . .
```

Q. 5 Write the program to search an element using linear search as well as binary search.

```c
#include <stdio.h>

// Linear search function
int linearSearch(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i; // Element found, return its index
        }
    }
    return -1; // Element not found, return -1
}

// Binary search function (requires a sorted array)
int binarySearch(int arr[], int size, int target) {
    int left = 0, right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Element found, return its index
        } else if (arr[mid] < target) {
            left = mid + 1; // Search in the right half
        } else {
            right = mid - 1; // Search in the left half
```

```
        }
    }

    return -1; // Element not found, return -1
}

int main() {
    int size, target;

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

    int arr[size];

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

    printf("Enter the element to search: ");
    scanf("%d", &target);

    // Perform linear search
    int linearIndex = linearSearch(arr, size, target);
```

```
    scanf("%d", &target);

    // Perform linear search
    int linearIndex = linearSearch(arr, size, target);
    if (linearIndex != -1) {
        printf("Linear Search: Element found at index %d\n", linearIndex);
    } else {
        printf("Linear Search: Element not found\n");
    }

    // Perform binary search
    int binaryIndex = binarySearch(arr, size, target);
    if (binaryIndex != -1) {
        printf("Binary Search: Element found at index %d\n", binaryIndex);
    } else {
        printf("Binary Search: Element not found\n");
    }

    return 0;
}
```

```
Enter the size of the array: 5
Enter 5 sorted elements:
10
40
60
80
70
Enter the element to search: 80
Linear Search: Element found at index 3
Binary Search: Element found at index 3

--------------------------------
Process exited after 10.62 seconds with return value 0
Press any key to continue . . .
```

Q. 6 Take an array of 20 integer inputs from user and print the following:

a. number of positive numbers

b. number of negative numbers

c. number of odd numbers

d. number of even numbers

e. number of 0.

```c
#include <stdio.h>

int main() {
    const int size = 20;
    int arr[size];

    // Input from the user
    printf("Enter %d integer elements:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Initialize counters
    int positiveCount = 0, negativeCount = 0, oddCount = 0, evenCount = 0, zeroCount = 0;

    // Counting loop
    for (int i = 0; i < size; i++) {
        // Count positive numbers
        if (arr[i] > 0) {
            positiveCount++;
        }
        // Count negative numbers
        else if (arr[i] < 0) {
            negativeCount++;
        }
```

Ans.

```c
        else if (arr[i] < 0) {
            negativeCount++;
        }
        // Count zeros
        else {
            zeroCount++;
        }

        // Count odd and even numbers
        if (arr[i] % 2 == 0) {
            evenCount++;
        } else {
            oddCount++;
        }
    }

    // Output the results
    printf("a. Number of positive numbers: %d\n", positiveCount);
    printf("b. Number of negative numbers: %d\n", negativeCount);
    printf("c. Number of odd numbers: %d\n", oddCount);
    printf("d. Number of even numbers: %d\n", evenCount);
    printf("e. Number of zeros: %d\n", zeroCount);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
- Output Size: 128.6015625 KiB
- Compilation Time: 0.34s
```

Sel: 0 Lines: 47 Length: 1221 Insert Done parsing in 0.016 seconds

C:\Users\hp\Desktop\C progr

```
Enter 20 integer elements:
10
202
30
25
14
5
5
8
5
4
7
98
26
54
87
15
46
26
2
8
a. Number of positive numbers: 20
b. Number of negative numbers: 0
c. Number of odd numbers: 7
d. Number of even numbers: 13
e. Number of zeros: 0

--------------------------------
Process exited after 40.25 seconds with return value 0
Press any key to continue . . .
```

Q. 7 Take an array of 10 elements. Split it into middle and store the elements in two different arrays.

E.g.-

INITIAL array: 58, 24, 13, 15, 63, 9, 8, 81, 1, 78

After splitting:

58, 24, 13, 15, 63

9, 8, 81, 1, 78

```c
#include <stdio.h>

int main() {
    const int size = 10;
    int initialArray[size], firstHalf[size / 2], secondHalf[size / 2];

    // Input from the user
    printf("Enter %d integer elements:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &initialArray[i]);
    }

    // Split the array into two halves
    for (int i = 0; i < size / 2; i++) {
        firstHalf[i] = initialArray[i];
        secondHalf[i] = initialArray[i + size / 2];
    }

    // Print the initial array
    printf("INITIAL array:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", initialArray[i]);
    }
    printf("\n");

```

Ans.

```c
    // Print the initial array
    printf("INITIAL array:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", initialArray[i]);
    }
    printf("\n");

    // Print the split arrays
    printf("After splitting:\n");
    for (int i = 0; i < size / 2; i++) {
        printf("%d ", firstHalf[i]);
    }
    printf("\n");
    for (int i = 0; i < size / 2; i++) {
        printf("%d ", secondHalf[i]);
    }
    printf("\n");

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter 10 integer elements:
58
24
13
15
63
9
8
81
1
78
INITIAL array:
58 24 13 15 63 9 8 81 1 78
After splitting:
58 24 13 15 63
9 8 81 1 78

--------------------------------
Process exited after 47.72 seconds with return value 0
Press any key to continue . . .
```

Q. 8 Write the program to count frequency of each element in an array.

Ans.

```c
#include <stdio.h>

void countFrequency(int arr[], int size) {

    int frequency[size];
    for (int i = 0; i < size; i++) {
        frequency[i] = -1;
    }
    for (int i = 0; i < size; i++) {
        int count = 1;

        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                count++;
                frequency[j] = 0;
            }
        }

        if (frequency[i] != 0) {
            frequency[i] = count;
        }
    }
    printf("Element\tFrequency\n");
    for (int i = 0; i < size; i++) {
        if (frequency[i] != 0) {
            printf("%d\t%d\n", arr[i], frequency[i]);
        }
    }
}

int main() {
    const int size = 10;
    int arr[size];
    printf("Enter %d integer elements:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    countFrequency(arr, size);

    return 0;
}
```

```
Compile Log  Debug  Find Results  Close
Compilation results...
--------
```

```
C:\Users\hp\Desktop\C progr
Enter 10 integer elements:
10
20
30
50
40
20
10
30
02
40
Element Frequency
10      2
20      2
30      2
50      1
40      2
2       1

--------------------------------
Process exited after 16.66 seconds w
Press any key to continue . . .
```

Week -7

Q. 1 Write the program to print row major and column major matrix.

```c
#include <stdio.h>

void printRowMajor(int matrix[][3], int rows, int cols)
 {
    printf("Row Major Order:\n");
    for (int i = 0; i < rows; ++i)
    {
        for (int j = 0; j < cols; ++j)
        {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void printColumnMajor(int matrix[][3], int rows, int cols)
{
    printf("Column Major Order:\n");
    for (int j = 0; j < cols; ++j)
     {
        for (int i = 0; i < rows; ++i)
        {
            printf("%d ", matrix[i][j]);
        }
```

Compile Log | Debug | Find Results | Close

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

int main()
 {
    int rows = 3;
    int cols = 3;

    int matrix[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    printRowMajor(matrix, rows, cols);
    printf("\n");
    printColumnMajor(matrix, rows, cols);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Row Major Order:
1 2 3
4 5 6
7 8 9

Column Major Order:
1 4 7
2 5 8
3 6 9

--------------------------------
Process exited after 0.06503 seconds with return value 0
Press any key to continue . . .
```

Q. 2 Write the program to print sum of a whole matrix.

```c
#include <stdio.h>

int calculateMatrixSum(int matrix[][3], int rows, int cols)
 {
    int sum = 0;

    for (int i = 0; i < rows; ++i)
    {
        for (int j = 0; j < cols; ++j)
        {
            sum += matrix[i][j];
        }
    }

    return sum;
}

int main()
{
    int rows = 3;
    int cols = 3;

    int matrix[3][3] =
     {
        {1, 2, 3},
```

```c
{
    int rows = 3;
    int cols = 3;

    int matrix[3][3] =
     {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int sum = calculateMatrixSum(matrix, rows, cols);

    printf("Sum of the matrix: %d\n", sum);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Sum of the matrix: 45

--------------------------------
Process exited after 0.06276 seconds with return value 0
Press any key to continue . . .
```

Q. 3 Write a program to add and multiply two 3x3 matrices. You can use 2D array to create a matrix.

```c
#include <stdio.h>

void addMatrices(int mat1[3][3], int mat2[3][3], int result[3][3])
 {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            result[i][j] = mat1[i][j] + mat2[i][j];
}

void multiplyMatrices(int mat1[3][3], int mat2[3][3], int result[3][3])
{
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++) {
            result[i][j] = 0;
            for (int k = 0; k < 3; k++)
                result[i][j] += mat1[i][k] * mat2[k][j];
        }
}

void displayMatrix(int mat[3][3])
 {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++)
            printf("%d\t", mat[i][j]);
        printf("\n");
    }
}

int main()
 {
    int mat1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int mat2[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int resultAddition[3][3], resultMultiplication[3][3];

    // Add matrices
```

Compile Log | Debug | Find Results | Close

```c
    // Add matrices
    addMatrices(mat1, mat2, resultAddition);

    // Multiply matrices
    multiplyMatrices(mat1, mat2, resultMultiplication);

    // Display results
    printf("Matrix 1:\n");
    displayMatrix(mat1);

    printf("\nMatrix 2:\n");
    displayMatrix(mat2);

    printf("\nMatrix Addition:\n");
    displayMatrix(resultAddition);

    printf("\nMatrix Multiplication:\n");
    displayMatrix(resultMultiplication);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
```

C:\Users\hp\Desktop\C progr

```
Matrix 1:
1       2       3
4       5       6
7       8       9

Matrix 2:
9       8       7
6       5       4
3       2       1

Matrix Addition:
10      10      10
10      10      10
10      10      10

Matrix Multiplication:
30      24      18
84      69      54
138     114     90

--------------------------------
Process exited after 0.06805 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Write the program to print sum of all diagonal elements, upper triangular matrix and lower triangular matrix.

```c
#include <stdio.h>

#define SIZE 3 // Assuming a square matrix of size 3x3

void printMatrix(int mat[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%d\t", mat[i][j]);
        }
        printf("\n");
    }
}

void sumDiagonalElements(int mat[SIZE][SIZE]) {
    int sum = 0;
    for (int i = 0; i < SIZE; i++) {
        sum += mat[i][i]; // Diagonal elements have the same row and column index
    }
    printf("Sum of diagonal elements: %d\n", sum);
}

void upperTriangularMatrix(int mat[SIZE][SIZE]) {
    printf("Upper Triangular Matrix:\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (j >= i) {
                printf("%d\t", mat[i][j]);
            } else {
                printf("0\t");
            }
        }
        printf("\n");
```

Compile Log | Debug | Find Results | Close

```c
}

void lowerTriangularMatrix(int mat[SIZE][SIZE]) {
    printf("Lower Triangular Matrix:\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (j <= i) {
                printf("%d\t", mat[i][j]);
            } else {
                printf("0\t");
            }
        }
        printf("\n");
    }
}

int main() {
    int matrix[SIZE][SIZE] = {{1, 2, 3},
                              {4, 5, 6},
                              {7, 8, 9}};

    printf("Original Matrix:\n");
    printMatrix(matrix);

    sumDiagonalElements(matrix);

    upperTriangularMatrix(matrix);

    lowerTriangularMatrix(matrix);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

Compilation results...

```
C:\Users\hp\Desktop\C progr
Original Matrix:
1       2       3
4       5       6
7       8       9
Sum of diagonal elements: 15
Upper Triangular Matrix:
1       2       3
0       5       6
0       0       9
Lower Triangular Matrix:
1       0       0
4       5       0
7       8       9

--------------------------------
Process exited after 0.1163 seconds with return value 0
Press any key to continue . . .
```

Q. 5 Write the program to find the frequency of odd and even elements in matrix.

```c
#include <stdio.h>
#define ROWS 3
#define COLS 3

void findFrequency(int mat[ROWS][COLS]) {
    int oddCount = 0, evenCount = 0;

    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++) {
            if (mat[i][j] % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }
        }
    }

    printf("Frequency of Odd Elements: %d\n", oddCount);
    printf("Frequency of Even Elements: %d\n", evenCount);
}

void displayMatrix(int mat[ROWS][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
```

Compile Log | Debug | Find Results | Close

Compilation results...

```c
void displayMatrix(int mat[ROWS][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%d\t", mat[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int matrix[ROWS][COLS] = {{1, 2, 3},
                              {4, 5, 6},
                              {7, 8, 9}};

    printf("Original Matrix:\n");
    displayMatrix(matrix);

    findFrequency(matrix);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Original Matrix:
1       2       3
4       5       6
7       8       9
Frequency of Odd Elements: 5
Frequency of Even Elements: 4

--------------------------------
Process exited after 0.06702 seconds with return value 0
Press any key to continue . . .
```

Q. 6 Write the program to find sum of each row and sum of each column of matrix.

```c
#include <stdio.h>

#define ROWS 3
#define COLS 3

void calculateSum(int mat[ROWS][COLS]) {
    int rowSum[ROWS] = {0};
    int colSum[COLS] = {0};

    // Calculate sum of each row and each column
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            rowSum[i] += mat[i][j];
            colSum[j] += mat[i][j];
        }
    }

    // Print sum of each row
    printf("Sum of Each Row:\n");
    for (int i = 0; i < ROWS; i++) {
        printf("Row %d: %d\n", i + 1, rowSum[i]);
    }

    // Print sum of each column
    printf("Sum of Each Column:\n");
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
```

```c
    printf("Sum of Each Column:\n");
    for (int j = 0; j < COLS; j++) {
        printf("Column %d: %d\n", j + 1, colSum[j]);
    }
}

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

int main() {
    int matrix[ROWS][COLS] = {{1, 2, 3},
                              {4, 5, 6},
                              {7, 8, 9}};

    printf("Original Matrix:\n");
    displayMatrix(matrix);

    calculateSum(matrix);

    return 0;
}
```

Resources | Compile Log | Debug | Find Results | Close

```
C:\Users\hp\Desktop\C progr
Original Matrix:
1       2       3
4       5       6
7       8       9
Sum of Each Row:
Row 1: 6
Row 2: 15
Row 3: 24
Sum of Each Column:
Column 1: 12
Column 2: 15
Column 3: 18

--------------------------------
Process exited after 0.05349 seconds with return value 0
Press any key to continue . . .
```

Q. 7 Initialize a 2D array of 3*3 matrix. E.g.-

1 2 3

2 3 4

3 4 5

```c
#include <stdio.h>

#define ROWS 3
#define COLS 3

int main() {
    int matrix[ROWS][COLS] = {{1, 2, 3},
                              {2, 3, 4},
                              {3, 4, 5}};

    // Print the initialized matrix
    printf("Initialized 3x3 Matrix:\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Initialized 3x3 Matrix:
1       2       3
2       3       4
3       4       5

--------------------------------
Process exited after 0.06559 seconds with return value 0
Press any key to continue . . .
```

Q. 8 A square matrix, one having the same number of rows and columns, is called a diagonal matrix if it's only non-zero elements are on the diagonal from upper left to lower right. It is called upper triangular matrix if all elements bellow the diagonal are zeroes, and lower triangular matrix, if all the elements above the diagonal are zeroes. Write a program that reads a matrix and determines if it is one of the above mentioned three special matrices.

```c
#include <stdio.h>

#define SIZE 3 // Assuming a square matrix of size 3x3

int isDiagonalMatrix(int mat[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (i != j && mat[i][j] != 0) {
                return 0; // Not a diagonal matrix
            }
        }
    }
    return 1; // Diagonal matrix
}

int isUpperTriangularMatrix(int mat[SIZE][SIZE]) {
    for (int i = 1; i < SIZE; i++) {
        for (int j = 0; j < i; j++) {
            if (mat[i][j] != 0) {
                return 0; // Not an upper triangular matrix
            }
        }
    }
    return 1; // Upper triangular matrix
}
```

```c

int isLowerTriangularMatrix(int mat[SIZE][SIZE]) {
    for (int i = 0; i < SIZE - 1; i++) {
        for (int j = i + 1; j < SIZE; j++) {
            if (mat[i][j] != 0) {
                return 0; // Not a lower triangular matrix
            }
        }
    }
    return 1; // Lower triangular matrix
}

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

int main() {
    int matrix[SIZE][SIZE];

    // Reading the matrix
```

```c
    // Reading the matrix
    printf("Enter a %dx%d matrix:\n", SIZE, SIZE);
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display the entered matrix
    printf("Entered Matrix:\n");
    displayMatrix(matrix);

    // Check and display the type of matrix
    if (isDiagonalMatrix(matrix)) {
        printf("It is a Diagonal Matrix.\n");
    } else if (isUpperTriangularMatrix(matrix)) {
        printf("It is an Upper Triangular Matrix.\n");
    } else if (isLowerTriangularMatrix(matrix)) {
        printf("It is a Lower Triangular Matrix.\n");
    } else {
        printf("It is not one of the special matrices.\n");
    }

    return 0;
}
```

Compile Log | Debug | Find Results | Close

Compilation results...

```
C:\Users\hp\Desktop\C progr
Enter a 3x3 matrix:
1
0
0
0
1
0
0
0
1
Entered Matrix:
1       0       0
0       1       0
0       0       1
It is a Diagonal Matrix.

--------------------------------
Process exited after 17.22 seconds with return value 0
Press any key to continue . . .
```

Q. 9 Write the program to check whether the matrix is sparse matrix or not.

```c
#include <stdio.h>
#define ROWS 3
#define COLS 3

int isSparseMatrix(int mat[ROWS][COLS]) {
    int zeroCount = 0;

    // Count the number of zeros in the matrix
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            if (mat[i][j] == 0) {
                zeroCount++;
            }
        }
    }

    // If the number of zeros is more than half of the total elements, it is considered sparse
    if (zeroCount > (ROWS * COLS) / 2) {
        return 1; // Sparse matrix
    } else {
        return 0; // Not a sparse matrix
    }
}

void displayMatrix(int mat[ROWS][COLS]) {
```

Compile Log | Debug | Find Results | Close

```c

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

int main() {
    int matrix[ROWS][COLS];

    // Reading the matrix
    printf("Enter a %dx%d matrix:\n", ROWS, COLS);
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display the entered matrix
    printf("Entered Matrix:\n");
    displayMatrix(matrix);

```

urces | Compile Log | Debug | Find Results | Close

```c
        for (int j = 0; j < COLS; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display the entered matrix
    printf("Entered Matrix:\n");
    displayMatrix(matrix);

    // Check and display whether the matrix is sparse or not
    if (isSparseMatrix(matrix)) {
        printf("It is a Sparse Matrix.\n");
    } else {
        printf("It is not a Sparse Matrix.\n");
    }

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

```
C:\Users\hp\Desktop\C progr
Enter a 3x3 matrix:
1
0
2
0
0
0
4
5
0
Entered Matrix:
1       0       2
0       0       0
4       5       0
It is a Sparse Matrix.

--------------------------------
Process exited after 15.73 seconds with return value 0
Press any key to continue . . .
```

Week -8

Q. 1 Write a C program to create, initialize and use pointers.

```c
#include <stdio.h >
int main ()
{
    int a=10;
    int *p=&a;
    printf(" Valu of pointer : %d",*p);
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
 Valu of pointer : 10
--------------------------------
Process exited after 0.05358 seconds with return value 0
Press any key to continue . . .
```

Q. 2 Write a C program to add two numbers using pointers.

```c
#include <stdio.h >
int main ()
{
    int a=10,b=20;
    int *p=&a;
    int *q=&b;
    int c=*p+*q;
    printf("Sum of pointer : %d",c);
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Sum of pointer : 30
--------------------------------
Process exited after 0.05334 seconds with return value 0
Press any key to continue . . .
```

Q. 3 Write a C program to swap two numbers using pointers.

```cpp
#include <stdio.h >
int main ()
{
    int a=10,b=20;
    int *p=&a;
    int *q=&b;
    printf("\n before swapping *p=%d,*q=%d",*p,*q);
    int c=*q;
     *q=*p;
    printf("\n After swapping :*p=%d, *q=%d",c,*q);
}
```

```
 before swapping *p=10,*q=20
 After swapping :*p=20, *q=10
--------------------------------
Process exited after 0.08112 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Write a C program to input and print array elements using pointer.

```c
#include <stdio.h>
#define MAX_SIZE 10
int main() {
    int arr[MAX_SIZE];
    int *ptr;
    int size;
    printf("Enter the size of the array (up to %d): ", MAX_SIZE);
    scanf("%d", &size);

    if (size > MAX_SIZE || size <= 0) {
        printf("Invalid array size. Please enter a size between 1 and %d.\n", MAX_SIZE);
        return 1;
    }
    printf("Enter %d elements for the array:\n", size);
    for (ptr = arr; ptr < arr + size; ptr++) {
        scanf("%d", ptr);
    }
    printf("Array elements are:\n");
    for (ptr = arr; ptr < arr + size; ptr++) {
        printf("%d\t", *ptr);
    }
    return 0;
}
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
```

```
Enter the size of the array (up to 10): 5
Enter 5 elements for the array:
10
20
30
40
50
Array elements are:
10      20      30      40      50
--------------------------------
Process exited after 11.09 seconds with return value 0
Press any key to continue . . .
```

Q. 5 Write a C program to copy one array to another using pointer.

```c
#include <stdio.h>
#define MAX_SIZE 10
void copyArray(int *source, int *destination, int size) {
    for (int i = 0; i < size; i++) {
        *(destination + i) = *(source + i);
    }
}
void displayArray(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d\t", *(arr + i));
    }
    printf("\n");
}
int main() {
    int sourceArray[MAX_SIZE];
    int destinationArray[MAX_SIZE];
    int size;
    printf("Enter the size of the array (up to %d): ", MAX_SIZE);
    scanf("%d", &size);

    if (size > MAX_SIZE || size <= 0) {
        printf("Invalid array size. Please enter a size between 1 and %d.\n", MAX_SIZE);
        return 1;
    }
    printf("Enter %d elements for the source array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &sourceArray[i]);
    }
    copyArray(sourceArray, destinationArray, size);
    printf("Source Array:\n");
    displayArray(sourceArray, size);

    printf("Destination Array (after copying):\n");
    displayArray(destinationArray, size);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
```

```
C:\Users\hp\Desktop\C progr
Enter the size of the array (up to 10): 5
Enter 5 elements for the source array:
6
2
30
50
4
Source Array:
6       2       30      50      4
Destination Array (after copying):
6       2       30      50      4

--------------------------------
Process exited after 12.05 seconds with return
Press any key to continue . . .
```

Q. 6 Write a C program to swap two arrays using pointers.

```c
#include <stdio.h>

#define MAX_SIZE 10

void swapArrays(int *arr1, int *arr2, int size) {
    // Swap elements using pointers
    for (int i = 0; i < size; i++) {
        // Using a temporary variable to swap elements
        int temp = *(arr1 + i);
        *(arr1 + i) = *(arr2 + i);
        *(arr2 + i) = temp;
    }
}

void displayArray(int *arr, int size) {
    // Display array elements using pointers
    for (int i = 0; i < size; i++) {
        printf("%d\t", *(arr + i));
    }
    printf("\n");
}

int main() {
    int array1[MAX_SIZE];
    int array2[MAX_SIZE];
```

```c
    int size;
    printf("Enter the size of the arrays (up to %d): ", MAX_SIZE);
    scanf("%d", &size);
    if (size > MAX_SIZE || size <= 0) {
        printf("Invalid array size. Please enter a size between 1 and %d.\n", MAX_SIZE);
        return 1;
    }
    printf("Enter %d elements for the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }
    printf("Enter %d elements for the second array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array2[i]);
    }
    swapArrays(array1, array2, size);

    printf("Array 1 (after swapping):\n");
    displayArray(array1, size);

    printf("Array 2 (after swapping):\n");
    displayArray(array2, size);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
- Output Size: 130.5166015625 KiB
```

```
C:\Users\hp\Desktop\C progr
Enter the size of the arrays (up to 10): 4
Enter 4 elements for the first array:
10
40
20
50
Enter 4 elements for the second array:
20
10
30
70
Array 1 (after swapping):
20      10      30      70
Array 2 (after swapping):
10      40      20      50

--------------------------------
Process exited after 19.25 seconds with return value 0
Press any key to continue . . .
```

Q. 7 Write a C program to reverse an array using pointers.

```c
#include <stdio.h>

void reverseArray(int arr[], int size) {
    int *start = arr;
    int *end = arr + size - 1;

    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Original Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    reverseArray(arr, size);
    printf("\nReversed Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Original Array: 1 2 3 4 5
Reversed Array: 5 4 3 2 1
--------------------------------
Process exited after 0.05628 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

Compilation results...

Q. 8 Write a C program to add two matrix using pointers.

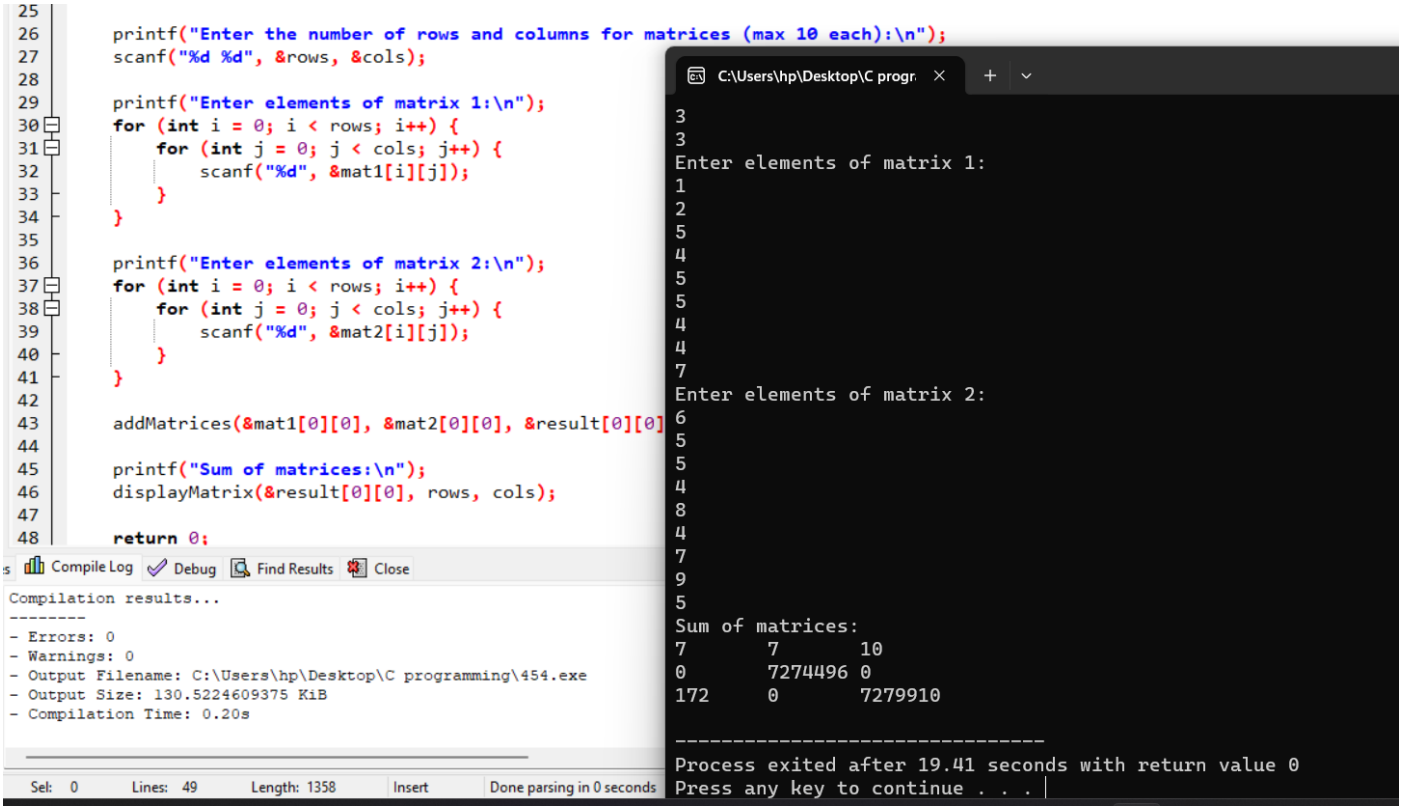
```c
#include <stdio.h>
#define MAX_ROWS 10
#define MAX_COLS 10

void addMatrices(int *mat1, int *mat2, int *result, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            *(result + i * cols + j) = *(mat1 + i * cols + j) + *(mat2 + i * cols + j);
        }
    }
}

void displayMatrix(int *matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", *(matrix + i * cols + j));
        }
        printf("\n");
    }
}

int main() {
    int mat1[MAX_ROWS][MAX_COLS], mat2[MAX_ROWS][MAX_COLS], result[MAX_ROWS][MAX_COLS];
    int rows, cols;

```



```
    printf("Enter the number of rows and columns for matrices (max 10 each):\n");
    scanf("%d %d", &rows, &cols);

    printf("Enter elements of matrix 1:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    printf("Enter elements of matrix 2:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    addMatrices(&mat1[0][0], &mat2[0][0], &result[0][0]

    printf("Sum of matrices:\n");
    displayMatrix(&result[0][0], rows, cols);

    return 0;
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
- Output Size: 130.5224609375 KiB
- Compilation Time: 0.20s
```

Sel: 0 Lines: 49 Length: 1358 Insert Done parsing in 0 seconds

C:\Users\hp\Desktop\C progr

```
3
3
Enter elements of matrix 1:
1
2
5
4
5
5
4
4
7
Enter elements of matrix 2:
6
5
5
4
8
4
7
9
5
Sum of matrices:
7       7       10
0       7274496 0
172     0       7279910

--------------------------------
Process exited after 19.41 seconds with return value 0
Press any key to continue . . .
```

Q. 9 Write a C program to multiply two matrix using pointers.

```c
#include <stdio.h>

#define MAX_ROWS1 10
#define MAX_COLS1 10
#define MAX_COLS2 10

void multiplyMatrices(int *mat1, int *mat2, int *result, int rows1, int cols1, int cols2) {
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            *(result + i * cols2 + j) = 0;
            for (int k = 0; k < cols1; k++) {
                *(result + i * cols2 + j) += *(mat1 + i * cols1 + k) * *(mat2 + k * cols2 + j);
            }
        }
    }
}

void displayMatrix(int *matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", *(matrix + i * cols + j));
        }
        printf("\n");
    }
}

int main() {
    int mat1[MAX_ROWS1][MAX_COLS1], mat2[MAX_COLS1][MAX_COLS2], result[MAX_ROWS1][MAX_COLS2];
    int rows1, cols1, rows2, cols2;

    printf("Enter the number of rows and columns for matrix 1 (max 10 each):\n");
    scanf("%d %d", &rows1, &cols1);
```

```c
    printf("Enter the number of rows and columns for matrix 1 (max 10 each):\n");
    scanf("%d %d", &rows1, &cols1);

    printf("Enter elements of matrix 1:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    printf("Enter the number of rows and columns for matrix 2 (max 10 rows and 10 columns):\n");
    scanf("%d %d", &rows2, &cols2);

    if (cols1 != rows2) {
        printf("Matrices cannot be multiplied. Columns of matrix 1 must be equal to rows of matrix 2.\n");
        return 1;
    }

    printf("Enter elements of matrix 2:\n");
    for (int i = 0; i < rows2; i++) {
        for (int j = 0; j < cols2; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    multiplyMatrices(&mat1[0][0], &mat2[0][0], &result[0][0], rows1, cols1, cols2);

    printf("Product of matrices:\n");
    displayMatrix(&result[0][0], rows1, cols2);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the number of rows and columns for matrix 1 (max 10 each):
2
2
Enter elements of matrix 1:
4
5
6
4
Enter the number of rows and columns for matrix 2 (max 10 rows and 10 columns):
2
2
Enter elements of matrix 2:
1
4
5
4
Product of matrices:
40985364        16
8257536 33030144

--------------------------------
Process exited after 17.61 seconds with return value 0
Press any key to continue . . .
```

Week – 9

Q. 1 Write a C program to Search string.

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

int searchString(char *mainString, char *subString) {
    int mainLen = strlen(mainString);
    int subLen = strlen(subString);

    for (int i = 0; i <= mainLen - subLen; i++) {
        int j;
        for (j = 0; j < subLen; j++) {
            if (mainString[i + j] != subString[j]) {
                break;
            }
        }
        if (j == subLen) {
            return i;
        }
    }

    return -1;
}
```

```c
int main() {
    char mainString[100], subString[50];

    printf("Enter the main string: ");
    fgets(mainString, sizeof(mainString), stdin);
    mainString[strcspn(mainString, "\n")] = '\0';

    printf("Enter the substring to search: ");
    fgets(subString, sizeof(subString), stdin);
    subString[strcspn(subString, "\n")] = '\0';
    int index = searchString(mainString, subString);
    if (index != -1) {
        printf("Substring found at index %d\n", index);
    } else {
        printf("Substring not found in the main string\n");
    }

    return 0;
}
```

```
Enter the main string: Pravendra Thakur
Enter the substring to search: d
Substring found at index 6

--------------------------------
Process exited after 23.85 seconds with return value 0
Press any key to continue . . .
```

Q. 2 Write a C program to count vowels, consonants, etc.

```c
#include <stdio.h>
#include <ctype.h>
int main() {
    char str[100];
    int vowels = 0, consonants = 0, digits = 0, spaces = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    for (int i = 0; str[i] != '\0'; i++) {
        if (isalpha(str[i])) {
            str[i] = tolower(str[i]);
            if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' || str[i] == 'o' || str[i] == 'u') {
                vowels++;
            }
            else {
                consonants++;
            }
        }
        else if (isdigit(str[i])) {
            digits++;
        }
        else if (str[i] == ' ') {
            spaces++;
        }
    }
    printf("Vowels: %d\n", vowels);
    printf("Consonants: %d\n", consonants);
    printf("Digits: %d\n", digits);
    printf("Spaces: %d\n", spaces);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a string: Pravendra
Vowels: 3
Consonants: 6
Digits: 0
Spaces: 0

--------------------------------
Process exited after 7.707 seconds with return value 0
Press any key to continue . . .
```

```
Compile Log  Debug  Find Results  Close
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

Q. 3 Create a program to separate characters in a given string?

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

int main() {
    char str[100];

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    printf("Separated characters: ");
    for (int i = 0; i < strlen(str); i++) {
        printf("%c ", str[i]);
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a string: HELLO
Separated characters: H E L L O

--------------------------------
Process exited after 6.786 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Write a program to take two strings from user and concatenate them also add a space between them using strcat() function.

Sample input: JAI

GLA

Sample output: JAI GLA

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str1[100];
    char str2[100];

    printf("Enter the first string: ");
    fgets(str1, sizeof(str1), stdin);

    printf("Enter the second string: ");
    fgets(str2, sizeof(str2), stdin);

    // Removing the newline character from fgets
    str1[strcspn(str1, "\n")] = '\0';
    str2[strcspn(str2, "\n")] = '\0';

    strcat(str1, " ");
    strcat(str1, str2);

    printf("Concatenated string: %s\n", str1);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first string: JAI
Enter the second string: GLA
Concatenated string: JAI GLA

--------------------------------
Process exited after 15.13 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

Compilation results...

Q. 5 Write a C program to take a string from user and make it toggle its case i.e. lower case to upper case and upper case to lower case.

Sample Input: HElLo wOrlD

Sample output: heLlO WoRLd

```c
#include <stdio.h>
#include <string.h>
void toggleCase(char str[]) {
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] >= 'a' && str[i] <= 'z') {
            str[i] = str[i] - 32; // converting lowercase to uppercase
        }
        else if (str[i] >= 'A' && str[i] <= 'Z') {
            str[i] = str[i] + 32; // converting uppercase to lowercase
        }
    }
}
int main() {
    char str[100];

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Removing the newline character from fgets
    str[strcspn(str, "\n")] = '\0';

    toggleCase(str);

    printf("Toggled case string: %s\n", str);
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a string: HElLowOrlD
Toggled case string: heLlOWoRLd

--------------------------------
Process exited after 54.99 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

Q. 6 Write a C program to take two strings as input from user and check they are identical or not without using string functions.

Sample input: Jai Gla

Jai Gla

Sample output: Identical

```c
#include <stdio.h>
#include<string.h>
int areStringsIdentical(char str1[], char str2[]) {
    int i = 0;
    while (str1[i] == str2[i]) {
        if (str1[i] == '\0' && str2[i] == '\0') {
            return 1; // strings are identical
        }
        i++;
    }
    return 0; // strings are not identical
}
int main() {
    char str1[100];
    char str2[100];

    printf("Enter the first string: ");
    fgets(str1, sizeof(str1), stdin);
    printf("Enter the second string: ");
    fgets(str2, sizeof(str2), stdin);
     str1[strcspn(str1, "\n")] = '\0';
    str2[strcspn(str2, "\n")] = '\0';

    if (areStringsIdentical(str1, str2)) {
        printf("The strings are identical.\n");
    } else {
        printf("The strings are not identical.\n");
    }
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first string: JAI GLA
Enter the second string: JAI GLA
The strings are identical.

--------------------------------
Process exited after 21.3 seconds with return value 0
Press any key to continue . . .
```

Week – 10

Q. 1 Write a C program to find length of string using pointers

```c
#include <stdio.h>

int stringLength(char *str) {
    int length = 0;

    while (*str != '\0') {
        length++;
        str++;
    }

    return length;
}

int main() {
    char str[100];

    printf("Enter a string: ");
    scanf("%s", str);

    int length = stringLength(str);

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

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a string: Pravendra
Length of the string: 9

--------------------------------
Process exited after 7.416 seconds with return value 0
Press any key to continue . . .
```

Compile Log Debug Find Results Close

Q. 2 Write a C program to copy one string to another using pointer.

```c
#include <stdio.h>

void stringCopy(char *source, char *destination) {
    while (*source != '\0') {
        *destination = *source;
        source++;
        destination++;
    }
    *destination = '\0';
}

int main() {
    char source[100], destination[100];

    printf("Enter a string: ");
    scanf("%s", source);

    stringCopy(source, destination);

    printf("Copied string: %s\n", destination);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a string: Pravendra
Copied string: Pravendra

--------------------------------
Process exited after 5.336 seconds with return value 0
Press any key to continue . . .
```

Q. 3 Write a C program to concatenate two strings using pointers.

```c
#include <stdio.h>

void stringConcatenate(char *str1, char *str2) {
    while (*str1 != '\0') {
        str1++;
    }

    while (*str2 != '\0') {
        *str1 = *str2;
        str1++;
        str2++;
    }

    *str1 = '\0';
}

int main() {
    char str1[100], str2[100];

    printf("Enter the first string: ");
    scanf("%s", str1);

    printf("Enter the second string: ");
    scanf("%s", str2);

    stringConcatenate(str1, str2);

    printf("Concatenated string: %s\n", str1);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first string: Pravendra
Enter the second string: Thakur
Concatenated string: PravendraThakur

--------------------------------
Process exited after 11.55 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Write a C program to compare two strings using pointers.

```c
int stringCompare(char *str1, char *str2) {
    while (*str1 == *str2) {
        if (*str1 == '\0')
            return 0;
        str1++;
        str2++;
    }
    return *str1 - *str2;
}

int main() {
    char str1[100], str2[100];

    printf("Enter the first string: ");
    scanf("%s", str1);

    printf("Enter the second string: ");
    scanf("%s", str2);

    int result = stringCompare(str1, str2);

    if (result == 0) {
        printf("Strings are equal\n");
    } else if (result < 0) {
        printf("First string is smaller\n");
    } else {
        printf("Second string is smaller\n");
    }

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first string: Pravendra
Enter the second string: Jasawat
Second string is smaller

--------------------------------
Process exited after 16.78 seconds with return value 0
Press any key to continue . . .
```

Q. 5 WAP to find largest among three numbers using pointer

```c
#include <stdio.h>

void findLargest(int *num1, int *num2, int *num3) {
    if (*num1 >= *num2 && *num1 >= *num3) {
        printf("%d is the largest number\n", *num1);
    } else if (*num2 >= *num1 && *num2 >= *num3) {
        printf("%d is the largest number\n", *num2);
    } else {
        printf("%d is the largest number\n", *num3);
    }
}

int main() {
    int num1, num2, num3;

    printf("Enter the first number: ");
    scanf("%d", &num1);

    printf("Enter the second number: ");
    scanf("%d", &num2);

    printf("Enter the third number: ");
    scanf("%d", &num3);

    findLargest(&num1, &num2, &num3);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first number: 5
Enter the second number: 6
Enter the third number: 8
8 is the largest number

--------------------------------
Process exited after 3.656 seconds with return value 0
Press any key to continue . . .
```

Q. 7 WAP to find factorial of a number using pointer.

```c
#include <stdio.h>

void findFactorial(int *num, unsigned long long *factorial) {
    *factorial = 1;

    for (int i = 1; i <= *num; i++) {
        *factorial *= i;
    }
}

int main() {
    int num;
    unsigned long long factorial;

    printf("Enter a number: ");
    scanf("%d", &num);

    findFactorial(&num, &factorial);

    printf("The factorial of %d is %llu\n", num, factorial);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter a number: 5
The factorial of 5 is 120

--------------------------------
Process exited after 4.91 seconds with return value 0
Press any key to continue . . .
```

Q. 8 Write a program to print largest even number present in an array using pointer to an array.

```c
#include <stdio.h>
int findLargestEven(int *arr, int size) {
    int largestEven = -1;

    for (int i = 0; i < size; i++) {
        if (*(arr + i) % 2 == 0 && *(arr + i) > largestEven) {
            largestEven = *(arr + i);
        }
    }

    return largestEven;
}

int main() {
    int arr[] = {5, 12, 8, 3, 18, 7, 10, 21};
    int size = sizeof(arr) / sizeof(arr[0]);

    int *ptr = arr;

    int largestEven = findLargestEven(ptr, size);

    if (largestEven != -1) {
        printf("The largest even number in the array is: %d\n", largestEven);
    } else {
        printf("There are no even numbers in the array.\n");
    }

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
```

```
C:\Users\hp\Desktop\C progr
The largest even number in the array is: 18

--------------------------------
Process exited after 0.03131 seconds with return value 0
Press any key to continue . . .
```

Q. 9 WAP to find sum of elements of an array using array of pointer.

```c
#include <stdio.h>

int findSum(int *arr[], int size) {
    int sum = 0;

    for (int i = 0; i < size; i++) {
        sum += *arr[i];
    }

    return sum;
}

int main() {
    int num1 = 5, num2 = 10, num3 = 15;
    int *arr[] = {&num1, &num2, &num3};
    int size = sizeof(arr) / sizeof(arr[0]);

    int sum = findSum(arr, size);

    printf("The sum of elements in the array is: %d\n", sum);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

```
C:\Users\hp\Desktop\C progr
The sum of elements in the array is: 30

--------------------------------
Process exited after 0.02954 seconds with return value 0
Press any key to continue . . .
```

Q. 10 WAP to compute simple interest using pointers.

```c
#include <stdio.h>

void computeSimpleInterest(float *principal, float *rate, float *time, float *interest) {
    *interest = (*principal * *rate * *time) / 100;
}

int main() {
    float principal, rate, time, interest;

    printf("Enter the principal amount: ");
    scanf("%f", &principal);

    printf("Enter the interest rate: ");
    scanf("%f", &rate);

    printf("Enter the time (in years): ");
    scanf("%f", &time);

    computeSimpleInterest(&principal, &rate, &time, &interest);

    printf("The simple interest is: %.2f\n", interest);

    return 0;
}
```

```
Compile Log  Debug  Find Results  Close
Compilation results...
--------
```

```
C:\Users\hp\Desktop\C progr
Enter the principal amount: 1500
Enter the interest rate: 2.5
Enter the time (in years): 3
The simple interest is: 112.50

--------------------------------
Process exited after 9.213 seconds with return value 0
Press any key to continue . . .
```

Q. 11 Write a program to print largest even number present in an array using pointer to an array.

```c
#include <stdio.h>
int findLargestEven(int *arr, int size) {
    int largestEven = -1;

    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0 && arr[i] > largestEven) {
            largestEven = arr[i];
        }
    }
    return largestEven;
}
int main() {
    int arr[] = {2, 5, 8, 12, 7, 10};
    int size = sizeof(arr) / sizeof(arr[0]);

    int largestEven = findLargestEven(arr, size);

    if (largestEven != -1) {
        printf("The largest even number in the array is: %d\n", largestEven);
    } else {
        printf("No even number found in the array.\n");
    }
    return 0;
}
```

```
Compile Log  Debug  Find Results  Close
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C programming\454.exe
- Output Size: 129.140625 KiB
```

```
C:\Users\hp\Desktop\C progr
The largest even number in the array is: 12

--------------------------------
Process exited after 0.0548 seconds with return value 0
Press any key to continue . . .
```

Week-11

Q. 1 Write a C function to return the maximum of three integers.

```c
#include <stdio.h>
int maxOfThree(int a, int b, int c) {
    int max = a;
    if (b > max) {
        max = b;
    }
    if (c > max) {
        max = c;
    }
    return max;
}

int main() {
    int num1, num2, num3;

    printf("Enter three integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    int maximum = maxOfThree(num1, num2, num3);

    printf("The maximum of the three integers is: %d\n", maximum);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter three integers: 25
23
143
The maximum of the three integers is: 143

--------------------------------
Process exited after 7.956 seconds with return value 0
Press any key to continue . . .
```

Q. 2 Write a C function to check if a given number is prime or not.

```c
#include <stdio.h>
int isPrime(int num) {
    if (num <= 1) {
        return 0;
    }
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return 0;
        }
    }
    return 1;
}
int main() {
    int number;

    printf("Enter a number: ");
    scanf("%d", &number);

    if (isPrime(number)) {
        printf("%d is a prime number.\n", number);
    } else {
        printf("%d is not a prime number.\n", number);
    }
    return 0;
```

```
C:\Users\hp\Desktop\C progr
Enter a number: 25
25 is not a prime number.

--------------------------------
Process exited after 2.281 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

Q. 3 Write a C function to compute the factorial of a non-negative integer.

```c
#include <stdio.h>

unsigned long long factorial(int num) {
    unsigned long long result = 1;

    for (int i = 1; i <= num; i++) {
        result *= i;
    }
    return result;
}
int main() {
    int number;
    printf("Enter a non-negative integer: ");
    scanf("%d", &number);
    if (number < 0) {
printf("Invalid input! Please enter a non-negative integer.\n");
        return 0;
    }
    unsigned long long fact = factorial(number);
    printf("The factorial of %d is %llu.\n", number, fact);
    return 0;
}
```

Close tab (Ctrl+F4)

```
C:\Users\hp\Desktop\C progr
Enter a non-negative integer: 5
The factorial of 5 is 120.

--------------------------------
Process exited after 4.158 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

Q. 4 Write a C function to swap the values of two integers in actual arguments.

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int num1, num2;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

    swap(&num1, &num2);

    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter two integers: 14
25
Before swapping: num1 = 14, num2 = 25
After swapping: num1 = 25, num2 = 14

--------------------------------
Process exited after 4.637 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
```

Q. 5 Write a C function to compute the sum and average of an array of integers.

```c
#include <stdio.h>
void computeSumAndAverage(int arr[], int size, int *sum, float *average) {
    *sum = 0;
    for (int i = 0; i < size; i++) {
        *sum += arr[i];
    }
    *average = (float)(*sum) / size;
}
int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    int sum;
    float average;
    computeSumAndAverage(arr, size, &sum, &average);
    printf("Sum: %d\n", sum);
    printf("Average: %.2f\n", average);
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the size of the array: 3
Enter the elements of the array:
20
10
30
Sum: 60
Average: 20.00

--------------------------------
Process exited after 8.099 seconds with return value 0
Press any key to continue . . .
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
```

Q. 6 Write a C function to find the GCD (Greatest Common Divisor) of two non-negative integers using Euclid's algori

```c
#include <stdio.h>

int findGCD(int num1, int num2) {
    // Base case: if num2 is 0, the GCD is num1
    if (num2 == 0) {
        return num1;
    }
    return findGCD(num2, num1 % num2);
}

int main() {
    int num1, num2;
    printf("Enter the first non-negative integer: ");
    scanf("%d", &num1);
    printf("Enter the second non-negative integer: ");
    scanf("%d", &num2);
    int gcd = findGCD(num1, num2);
    printf("The GCD of %d and %d is %d\n", num1, num2, gcd);
    return 0;
}
```

```
C:\Users\hp\Desktop\C progr
Enter the first non-negative integer: 20
Enter the second non-negative integer: 10
The GCD of 20 and 10 is 10

--------------------------------
Process exited after 8.352 seconds with return value 0
Press any key to continue . . .
```

Q. 7 Write a C function to check if a given string is a valid palindrome, considering only alphanumeric characters and ignoring cases.

```c
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

bool isPalindrome(char *str) {
    int left = 0;
    int right = strlen(str) - 1;

    while (left < right) {
        // Ignore non-alphanumeric characters
        while (!isalnum(str[left])) {
            left++;
        }

        while (!isalnum(str[right])) {
            right--;
        }

        // Ignore case sensitivity
        if (tolower(str[left]) != tolower(str[right])) {
            return false;
        }

        left++;
```

```c
        right--;
    }

    return true;
}

int main() {
    char str[100];

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Remove the newline character from fgets
    str[strcspn(str, "\n")] = '\0';

    if (isPalindrome(str)) {
        printf("The string is a valid palindrome.\n");
    } else {
        printf("The string is not a valid palindrome.\n");
    }

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
C:\Users\hp\Desktop\C progr
Enter a string: 121
The string is a valid palindrome.

--------------------------------
Process exited after 2.181 seconds with return value 0
Press any key to continue . . .
```

Q. 8 Write a C function to calculate the sum and difference of two complex numbers.

```c
#include <stdio.h>

typedef struct {
    float real;
    float imaginary;
} Complex;

Complex addComplex(Complex num1, Complex num2) {
    Complex result;
    result.real = num1.real + num2.real;
    result.imaginary = num1.imaginary + num2.imaginary;
    return result;
}

Complex subtractComplex(Complex num1, Complex num2) {
    Complex result;
    result.real = num1.real - num2.real;
    result.imaginary = num1.imaginary - num2.imaginary;
    return result;
}

int main() {
    Complex num1, num2, sum, difference;

```

Compile Log | Debug | Find Results | Close

```c
int main() {
    Complex num1, num2, sum, difference;

    printf("Enter the real and imaginary parts of the first complex number: ");
    scanf("%f %f", &num1.real, &num1.imaginary);

    printf("Enter the real and imaginary parts of the second complex number: ");
    scanf("%f %f", &num2.real, &num2.imaginary);

    sum = addComplex(num1, num2);
    difference = subtractComplex(num1, num2);

    printf("Sum: %.2f + %.2fi\n", sum.real, sum.imaginary);
    printf("Difference: %.2f + %.2fi\n", difference.real, difference.imaginary);

    return 0;
}
```

Compile Log | Debug | Find Results

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Desktop\C
- Output Size: 129.1904296875 KiB
- Compilation Time: 0.34s
```

C:\Users\hp\Desktop\C progr

```
Enter the real and imaginary parts of the first complex number: 124
215
Enter the real and imaginary parts of the second complex number: -147
-345
Sum: -23.00 + -130.00i
Difference: 271.00 + 560.00i

--------------------------------
Process exited after 25.01 seconds with return value 0
Press any key to continue . . .
```