WEEK 11

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

```
#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;
}
```

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

```
#include <stdio.h>
int isPrime(int num) {
   if (num <= 1) {
       return 0; // Not a prime number
   }
```

```
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return 0; // Not a prime number
        }
    }
    return 1; // Prime number
}

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;
}
```

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

```
#include <stdio.h>

unsigned long long factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    unsigned long long fact = 1;
    for (int i = 2; i <= n; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    int number;
    printf("Enter a non-negative integer: ");
    scanf("%d", &number);
    printf("Factorial of %d is: %llu\n", number, factorial(number));
    return 0;
}
```

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

```
#include <stdio.h>
void swapIntegers(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);
    swapIntegers(&num1, &num2);
    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}
```

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

```
#include <stdio.h>

void sumAndAverage(int arr[], int size, int *sum, double *average) {
    *sum = 0;
    for (int i = 0; i < size; i++) {
        *sum += arr[i];
    }
    *average = (double)(*sum) / size;
}

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

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

    int sum;
    double average;
    sumAndAverage(array, size, &sum, &average);
    printf("Sum of the array elements is: %d\n", sum);
    printf("Average of the array elements is: %.2f\n", average);
    return 0;
}
```

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

```
#include <stdio.h>

int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int num1, num2;
    printf("Enter two non-negative integers: ");
    scanf("%d %d", &num1, &num2);

    int greatestCommonDivisor = gcd(num1, num2);
    printf("GCD of %d and %d is: %d\n", num1, num2, greatestCommonDivisor);

    return 0;
}
```

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

```
#include <string.h>

int isAlphanumeric(char ch) {
   return (isalpha(ch) || isdigit(ch));
}

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

   while (left < right) {
      while (!isAlphanumeric(str[left])) {
         left++;
      }
      while (!isAlphanumeric(str[right])) {
         right--;
      }
      if (tolower(str[left]) != tolower(str[right])) {
         return 0; // Not a palindrome
      }
      left++;
      right--;
   }
   return 1; // Palindrome
}

int main() {
   char input[100];
   printf("Enter a string: ");
```

```
    fgets(input, sizeof(input), stdin);

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



    if (isPalindrome(input)) {

        printf("The given string is a palindrome.\n");

    } else {

        printf("The given string is not a palindrome.\n");

    }



    return 0;

}
```

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

```
#include <stdio.h>



struct Complex {

    float real;

    float imag;

};



void sumDifferenceComplex(struct Complex num1, struct Complex num2, struct Complex
*sum, struct Complex *diff) {

    sum->real = num1.real + num2.real;

    sum->imag = num1.imag + num2.imag;



    diff->real = num1.real - num2.real;

    diff->imag = num1.imag - num2.imag;

}



int main() {
    struct Complex number1, number2, sum, difference;

    printf("For first complex number:\n");
    printf("Enter real and imaginary parts respectively: ");
    scanf("%f %f", &number1.real, &number1.imag);

    printf("For second complex number:\n");
    printf("Enter real and imaginary parts respectively: ");
    scanf("%f %f", &number2.real, &number2.imag);

    sumDifferenceComplex(number1, number2, &sum, &difference);

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

    return 0;
}
```

WEEK 10:---

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

```
#include <stdio.h>

int stringLength(char *str) {
    int length = 0;
    while (*str != '\0') {
        length++;
```

```c
        str++;
    }
    return length;
}

int main() {
    char inputString[100];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets input

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

    return 0;
}
```

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

```c
#include <stdio.h>

void stringCopy(char *source, char *destination) {
    while (*source != '\0') {
        *destination = *source;
        source++;
        destination++;
    }
    *destination = '\0'; // Append null character at the end of the destination string
}
```

```
int main() {
    char sourceString[100], destinationString[100];
    printf("Enter a string to copy: ");
    fgets(sourceString, sizeof(sourceString), stdin);
    sourceString[strcspn(sourceString, "\n")] = '\0'; // Remove the newline character from fgets input

    stringCopy(sourceString, destinationString);
    printf("Copied string: %s\n", destinationString);

    return 0;
}
```

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

```
#include <stdio.h>

void stringConcatenate(char *str1, char *str2) {
    while (*str1 != '\0') {
        str1++;
    }
    while (*str2 != '\0') {
        *str1 = *str2;
        str1++;
        str2++;
    }
    *str1 = '\0'; // Append null character at the end of the concatenated string
}

int main() {
    char string1[100], string2[50];
    printf("Enter first string: ");
    fgets(string1, sizeof(string1), stdin);
    string1[strcspn(string1, "\n")] = '\0'; // Remove the newline character from fgets input

    printf("Enter second string: ");
    fgets(string2, sizeof(string2), stdin);
    string2[strcspn(string2, "\n")] = '\0'; // Remove the newline character from fgets input

    stringConcatenate(string1, string2);
    printf("Concatenated string: %s\n", string1);

    return 0;
}
```

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

```
#include <stdio.h>

int stringCompare(char *str1, char *str2) {
    while (*str1 == *str2) {
        if (*str1 == '\0' || *str2 == '\0') {
            break;
        }
        str1++;
        str2++;
    }
    if (*str1 == '\0' && *str2 == '\0') {
        return 0; // Strings are equal
    } else {
        return (*str1 - *str2); // Return the ASCII difference
    }
}

int main() {
    char string1[100], string2[100];
    printf("Enter first string: ");
    fgets(string1, sizeof(string1), stdin);
    string1[strcspn(string1, "\n")] = '\0'; // Remove the newline character from fgets input

    printf("Enter second string: ");
    fgets(string2, sizeof(string2), stdin);
    string2[strcspn(string2, "\n")] = '\0'; // Remove the newline character from fgets input

    int result = stringCompare(string1, string2);
    if (result == 0) {
        printf("Strings are equal.\n");
    } else if (result < 0) {
        printf("String 1 is less than String 2.\n");
    } else {
        printf("String 1 is greater than String 2.\n");
    }

    return 0;
}
```

Q. 5: Write a C program to find the largest among three numbers using pointers.

```
#include <stdio.h>

void findLargest(int *num1, int *num2, int *num3, int *largest) {
    *largest = (*num1 > *num2) ? ((*num1 > *num3) ? *num1 : *num3) : ((*num2 > *num3)
? *num2 : *num3);
}

int main() {
    int number1, number2, number3, largest;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &number1, &number2, &number3);

    findLargest(&number1, &number2, &number3, &largest);
    printf("The largest number among %d, %d, and %d is: %d\n", number1, number2,
number3, largest);

    return 0;
}
```

Q. 6: Write a C program to find the factorial of a number using pointers.

```
#include <stdio.h>

void findFactorial(int num, unsigned long long *fact) {
    *fact = 1;
    for (int i = 1; i <= num; i++) {
        *fact *= i;
    }
}

int main() {
    int number;
    unsigned long long factorial;
    printf("Enter a number: ");
    scanf("%d", &number);
    findFactorial(number, &factorial);
    printf("Factorial of %d is: %llu\n", number, factorial);

    return 0;
}
```

Q. 7: Write a program to print the largest even number present in an array using a pointer to an array.

```
#include <stdio.h>

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

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

    int largestEven = largestEvenNumber(array, 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;
}
```

Q. 8: Write a program to find the sum of elements of an array using an array of pointers.

```
#include <stdio.h>

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

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

    int elements[size];
    int *pointers[size];
    printf("Enter %d elements in the array: ", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &elements[i]);
        pointers[i] = &elements[i];
    }

    int totalSum = sumOfArray(pointers, size);
    printf("Sum of elements in the array is: %d\n", totalSum);

    return 0;
}
```

Q. 9: Write a program to compute simple interest using pointers.

```
#include <stdio.h>

void calculateInterest(float *principal, float *rate, float *time, float *interest) {
    *interest = (*principal * *rate * *time) / 100.0;
}

int main() {
    float principal, rate, time, interest;
    printf("Enter principal amount: ");
    scanf("%f", &principal);

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

    printf("Enter time period (in years): ");
    scanf("%f", &time);
    calculateInterest(&principal, &rate, &time, &interest);
    printf("Simple interest is: %.2f\n", interest);

    return 0;
}
```

WEEK 09 :--

Absolutely! Here are solutions to each of the problems you've mentioned:

Q. 1: Write a C program to search a string.

```
#include <stdio.h>
int searchString(char *str, char *search) {
    int i, j, flag;
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == search[0]) {
            flag = 1;
            for (j = 0; search[j] != '\0'; j++) {
                if (str[i + j] != search[j]) {
                    flag = 0;
                    break;
                }
            }
            if (flag == 1)
                return i;
       }
```

```c
    }
    return -1;
}
int main() {
    char inputString[100], searchStr[20];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets input

    printf("Enter a string to search for: ");
    fgets(searchStr, sizeof(searchStr), stdin);
    searchStr[strcspn(searchStr, "\n")] = '\0'; // Remove the newline character from fgets input

    int index = searchString(inputString, searchStr);
    if (index != -1)
        printf("'%s' found at index %d in '%s'\n", searchStr, index, inputString);
    else
        printf("'%s' not found in '%s'\n", searchStr, inputString);

    return 0;
}
```

Q. 2: Write a C program to reverse words in a string.

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

void reverseWords(char *str) {
    int start = 0;
    for (int end = 0; str[end] != '\0'; end++) {
        if (str[end] == ' ') {
            for (int i = end - 1; i >= start; i--) {
                printf("%c", str[i]);
            }
            printf(" ");
            start = end + 1;
        }
    }
    for (int i = strlen(str) - 1; i >= start; i--) {
        printf("%c", str[i]);
    }
}


int main() {
    char inputString[100];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets input


    printf("Reverse of words in the string: ");
    reverseWords(inputString);


    return 0;
}
```

Q. 3: Write a C program to count vowels, consonants, etc. in a string.

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

```
void countVowelsConsonants(char *str) {
    int vowels = 0, consonants = 0, digits = 0, spaces = 0, others = 0;


    for (int i = 0; str[i] != '\0'; i++) {
        if (isalpha(str[i])) {
            char ch = tolower(str[i]);
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                vowels++;
            } else {
                consonants++;
            }
        } else if (isdigit(str[i])) {
            digits++;
        } else if (str[i] == ' ') {
            spaces++;
        } else {
            others++;
        }
    }


    printf("Vowels: %d\nConsonants: %d\nDigits: %d\nSpaces: %d\nOthers: %d\n", vowels,
consonants, digits, spaces, others);
}


int main() {
    char inputString[100];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets
input
```

```
    countVowelsConsonants(inputString);

    return 0;
}
```

Please let me know if you need more explanations or if you'd like to continue with the remaining questions!

User

continue

ChatGPT

Certainly! Here are solutions to the next set of problems:

Q. 4: Create a program to separate characters in a given string.

```
#include <stdio.h>

void separateCharacters(char *str) {
    printf("Individual characters in the string: ");
    for (int i = 0; str[i] != '\0'; i++) {
        printf("%c ", str[i]);
    }
    printf("\n");
}

int main() {
```

```c
    char inputString[100];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets input

    separateCharacters(inputString);

    return 0;
}
```

Q. 5: Write a program to take two strings from the user, concatenate them, and add a space between them using strcat() function.

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

int main() {
    char string1[50], string2[50];
    printf("Enter the first string: ");
    fgets(string1, sizeof(string1), stdin);
    string1[strcspn(string1, "\n")] = '\0'; // Remove the newline character from fgets input

    printf("Enter the second string: ");
    fgets(string2, sizeof(string2), stdin);
    string2[strcspn(string2, "\n")] = '\0'; // Remove the newline character from fgets input

    strcat(string1, " ");
    strcat(string1, string2);

    printf("Concatenated string with space: %s\n", string1);
    return 0;
}
```

Q. 6: Write a C program to take a string from the user and toggle its case.

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

void toggleCase(char *str) {
    for (int i = 0; str[i] != '\0'; i++) {
        if (islower(str[i])) {
            str[i] = toupper(str[i]);
        } else if (isupper(str[i])) {
            str[i] = tolower(str[i]);
        }
    }
}

int main() {
    char inputString[100];
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);
    inputString[strcspn(inputString, "\n")] = '\0'; // Remove the newline character from fgets input

    toggleCase(inputString);
    printf("Toggled case string: %s\n", inputString);

    return 0;
}
```

Q. 7: Write a C program to take two strings as input from the user and check if they are identical without using string functions.

```
#include <stdio.h>

int compareStrings(char *str1, char *str2) {
    while (*str1 != '\0' && *str2 != '\0') {
        if (*str1 != *str2) {
            return 0; // Not identical
        }
        str1++;
        str2++;
    }
    if (*str1 == '\0' && *str2 == '\0') {
        return 1; // Identical
    }
    return 0;
}

int main() {
    char string1[100], string2[100];
    printf("Enter the first string: ");
    fgets(string1, sizeof(string1), stdin);
    string1[strcspn(string1, "\n")] = '\0'; // Remove the newline character from fgets input

    printf("Enter the second string: ");
    fgets(string2, sizeof(string2), stdin);
    string2[strcspn(string2, "\n")] = '\0'; // Remove the newline character from fgets input

    if (compareStrings(string1, string2)) {
        printf("Strings are identical.\n");
    } else {
        printf("Strings are not identical.\n");
    }

    return 0;
}
```

ASSINGMENT 8:

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

```
#include <stdio.h>

int main() {
    int number = 10;
    int *ptr;

    ptr = &number;

    printf("Value of number: %d\n", number);
    printf("Address of number: %p\n", &number);
    printf("Value of pointer ptr: %p\n", ptr);
    printf("Value pointed by pointer ptr: %d\n", *ptr);

    return 0;
}
```

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

```
#include <stdio.h>

void addNumbers(int *a, int *b, int *sum) {
    *sum = *a + *b;
}
```

```
int main() {
    int num1, num2, sum;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

    addNumbers(&num1, &num2, &sum);
    printf("Sum of %d and %d is: %d\n", num1, num2, sum);

    return 0;
}
```

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

```
#include <stdio.h>

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

int main() {
    int num1, num2;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);
    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);
    swapNumbers(&num1, &num2);
    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}
```

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

```
#include <stdio.h>

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

    int array[size];

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

    printf("Array elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", *(array + i));
    }
    printf("\n");

    return 0;
}
```

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

```
#include <stdio.h>

void copyArray(int *src, int *dest, int size) {
    for (int i = 0; i < size; i++) {
        *(dest + i) = *(src + i);
    }
}

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

    int array1[size], array2[size];

    printf("Enter %d elements in the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }

    copyArray(array1, array2, size);

    printf("Copied array elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array2[i]);
    }
    printf("\n");
    return 0;
}
```

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

```
#include <stdio.h>

#define MAX_SIZE 100

void swapArrays(int *arr1, int *arr2, int size) {
    int temp[MAX_SIZE];

    for (int i = 0; i < size; i++) {
        temp[i] = *(arr1 + i);
        *(arr1 + i) = *(arr2 + i);
        *(arr2 + i) = temp[i];
    }
}

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

    int array1[MAX_SIZE], array2[MAX_SIZE];

    printf("Enter %d elements in the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }
    printf("Enter %d elements in the second array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array2[i]);
    }

    swapArrays(array1, array2, size);

    printf("After swapping:\n");
    printf("Array 1 elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array1[i]);
    }
    printf("\n");

    printf("Array 2 elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array2[i]);
    }
    printf("\n");

    return 0;
}
```

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

```
#include <stdio.h>

void reverseArray(int *arr, int size) {
    int *start = arr;
    int *end = arr + size - 1;
    int temp;
    while (start < end) {
        temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

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

    int array[size];

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

    reverseArray(array, size);

    printf("Reversed array elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
    return 0;
}
```

WEEK -8---

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

```
#include <stdio.h>

int main() {
    int number = 10;
    int *ptr;

    ptr = &number;

    printf("Value of number: %d\n", number);
    printf("Address of number: %p\n", &number);
    printf("Value of pointer ptr: %p\n", ptr);
    printf("Value pointed by pointer ptr: %d\n", *ptr);

    return 0;
}
```

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

```
#include <stdio.h>

void addNumbers(int *a, int *b, int *sum) {
    *sum = *a + *b;
}

int main() {
    int num1, num2, sum;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

    addNumbers(&num1, &num2, &sum);
    printf("Sum of %d and %d is: %d\n", num1, num2, sum);

    return 0;
}
```

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

```
#include <stdio.h>

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

int main() {
    int num1, num2;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

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

```c
    return 0;
}
```

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

```c
#include <stdio.h>

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

    int array[size];

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

    printf("Array elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", *(array + i));
    }
    printf("\n");

    return 0;
}
```

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

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

```
void copyArray(int *src, int *dest, int size) {
    for (int i = 0; i < size; i++) {
        *(dest + i) = *(src + i);
    }
}

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

    int array1[size], array2[size];

    printf("Enter %d elements in the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }

    copyArray(array1, array2, size);

    printf("Copied array elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array2[i]);
    }
    printf("\n");

    return 0;
}
```

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

```c
#include <stdio.h>

#define MAX_SIZE 100

void swapArrays(int *arr1, int *arr2, int size) {
    int temp[MAX_SIZE];

    for (int i = 0; i < size; i++) {
        temp[i] = *(arr1 + i);
        *(arr1 + i) = *(arr2 + i);
        *(arr2 + i) = temp[i];
    }
}

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

    int array1[MAX_SIZE], array2[MAX_SIZE];

    printf("Enter %d elements in the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }

    printf("Enter %d elements in the second array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array2[i]);
    }

    swapArrays(array1, array2, size);

    printf("After swapping:\n");
    printf("Array 1 elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array1[i]);
    }
    printf("\n");

    printf("Array 2 elements are: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", array2[i]);
    }
    printf("\n");

    return 0;
}
```

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

```
#include <stdio.h>

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

    while (start < end) {
        temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

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

    int array[size];

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

    reverseArray(array, size);

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

    return 0;
}
```

WEEK7 --

Q. 1: Write a program to print row-major and column-major matrices.

```
#include <stdio.h>

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

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

    printRowMajor(matrix, 3, 3);
    printColumnMajor(matrix, 3, 3);

    return 0;
}
```

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

```
#include <stdio.h>

int sumMatrix(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 matrix[3][3] = {
        {1, 2, 3},
        {2, 3, 4},
        {3, 4, 5}
    };
    int sum = sumMatrix(matrix, 3, 3);
    printf("Sum of the whole matrix: %d\n", sum);

    return 0;
}
```

Q. 3: Write a program to add and multiply two 3x3 matrices.

```
#include <stdio.h>

void addMatrices(int mat1[][3], int mat2[][3], int result[][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], int mat2[][3], int result[][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]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int matrix1[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int matrix2[3][3] = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
    };

    int sumMatrix[3][3], productMatrix[3][3];

    addMatrices(matrix1, matrix2, sumMatrix);
    multiplyMatrices(matrix1, matrix2, productMatrix);

    printf("Matrix 1:\n");
    displayMatrix(matrix1);
    printf("\nMatrix 2:\n");
    displayMatrix(matrix2);
    printf("\nSum of matrices:\n");
    displayMatrix(sumMatrix);
    printf("\nProduct of matrices:\n");
    displayMatrix(productMatrix);

    return 0;
}
```

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

```
#include <stdio.h>

void printSumDiagonal(int matrix[][3]) {
    int sumDiagonal = 0;
    printf("Diagonal elements: ");
    for (int i = 0; i < 3; i++) {
        sumDiagonal += matrix[i][i];
        printf("%d ", matrix[i][i]);
    }
    printf("\nSum of diagonal elements: %d\n", sumDiagonal);
}

void printUpperLowerTriangle(int matrix[][3]) {
    int sumUpper = 0, sumLower = 0;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (i > j) {
                sumLower += matrix[i][j];
            }
            if (j > i) {
                sumUpper += matrix[i][j];
            }
        }
    }
    printf("Sum of upper triangular matrix: %d\n", sumUpper);
    printf("Sum of lower triangular matrix: %d\n", sumLower);
}

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

    printSumDiagonal(matrix);
    printUpperLowerTriangle(matrix);

    return 0;
}
```

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

```
#include <stdio.h>
void findFrequency(int matrix[][3]) {
    int oddCount = 0, evenCount = 0;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (matrix[i][j] % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }
        }
    }

    printf("Frequency of even numbers: %d\n", evenCount);
    printf("Frequency of odd numbers: %d\n", oddCount);
}

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

    findFrequency(matrix);

    return 0;
}
```

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

```
#include <stdio.h>

void sumRowsColumns(int matrix[][3]) {
    int rowSum[3] = {0}, colSum[3] = {0};

    printf("Sum of each row:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            rowSum[i] += matrix[i][j];
        }
        printf("Row %d: %d\n", i + 1, rowSum[i]);
    }

    printf("\nSum of each column:\n");
    for (int j = 0; j < 3; j++) {
        for (int i = 0; i < 3; i++) {
            colSum[j] += matrix[i][j];
        }
        printf("Column %d: %d\n", j + 1, colSum[j]);
    }
}

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

    sumRowsColumns(matrix);

    return 0;
}
```

Q. 7: Initialize a 2D array of a 3x3 matrix:

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

```
void printMatrix(int matrix[][3]) {
    printf("Matrix:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

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

    printMatrix(matrix);

    return 0;
}
```

Q. 8: Write a program that reads a matrix and determines if it is a diagonal, upper triangular, or lower triangular matrix.

```
#include <stdio.h>

int checkSpecialMatrix(int matrix[][3]) {
    int isDiagonal = 1, isUpperTriangular = 1, isLowerTriangular = 1;

    // Check for diagonal matrix
```

```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (i != j && matrix[i][j] != 0) {
            isDiagonal = 0;
            break;
        }
    }
}

// Check for upper triangular matrix
for (int i = 1; i < 3; i++) {
    for (int j = 0; j < i; j++) {
        if (matrix[i][j] != 0) {
            isUpperTriangular = 0;
            break;
        }
    }
}

// Check for lower triangular matrix
for (int i = 0; i < 3; i++) {
    for (int j = i + 1; j < 3; j++) {
        if (matrix[i][j] != 0) {
            isLowerTriangular = 0;
            break;
        }
    }
}
if (isDiagonal) {
      return 1;
    } else if (isUpperTriangular) {
        return 2; // Upper triangular matrix
    } else if (isLowerTriangular) {
        return 3; // Lower triangular matrix
    } else {
        return 0; // None of the special matrices
    }
}

int main() {
    int matrix[3][3] = {
        {1, 0, 0},
        {0, 2, 0},
        {0, 0, 3}
    };

    int result = checkSpecialMatrix(matrix);

    if (result == 1) {
        printf("The matrix is a diagonal matrix.\n");
    } else if (result == 2) {
        printf("The matrix is an upper triangular matrix.\n");
    } else if (result == 3) {
        printf("The matrix is a lower triangular matrix.\n");
    } else {
        printf("The matrix is not a special matrix.\n");
    }
    return 0;
}
```

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

```
#include <stdio.h>
int checkSparseMatrix(int matrix[][3]) {
    int countZeroes = 0;

    // Count the number of zeroes in the matrix
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (matrix[i][j] == 0) {
                countZeroes++;
            }
        }
    }

    // If number of zeroes is more than half of the matrix elements, it is considered sparse
    if (countZeroes > (3 * 3) / 2) {
        return 1; // Sparse matrix
    } else {
        return 0; // Not a sparse matrix
    }
}

int main() {
    int matrix[3][3] = {
        {1, 0, 0},
        {0, 2, 0},
        {0, 0, 3}
    };

    int result = checkSparseMatrix(matrix);

    if (result) {
        printf("The matrix is a sparse matrix.\n");
    } else {
        printf("The matrix is not a sparse matrix.\n");
    }

    return 0;
}
```

WEEK 06 ---

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

```
#include <stdio.h>
void displayArray(int arr[], int size) {
    printf("Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n"); }
void insertElement(int arr[], int *size, int position, int element) {
    if (*size >= 100) {
        printf("Array is full. Cannot insert more elements.\n");
        return;
    }
    if (position < 0 || position > *size) {
        printf("Invalid position for insertion.\n");
        return;
    }
    for (int i = *size - 1; i >= position; i--) {
        arr[i + 1] = arr[i];
    }
    arr[position] = element;
    (*size)++;
}
void deleteElement(int arr[], int *size, int position) {
    if (*size <= 0) {
        printf("Array is empty. Cannot delete elements.\n");
        return;
    }
    if (position < 0 || position >= *size) {
        printf("Invalid position for deletion.\n");
        return;
    }
    for (int i = position; i < *size - 1; i++) {
        arr[i] = arr[i + 1];
    }
    (*size)--;
}
int main() {
    int arr[100], size = 0, choice, position, element;
    do {
        printf("\nMenu:\n");
        printf("1. Insert element\n");
        printf("2. Delete element\n");
        printf("3. Display array\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
          case 1:
            printf("Enter position to insert: ");
            scanf("%d", &position);
            printf("Enter element to insert: ");
            scanf("%d", &element);
            insertElement(arr, &size, position, element);
            break;
          case 2:
            printf("Enter position to delete: ");
            scanf("%d", &position);
            deleteElement(arr, &size, position);
            break;
          case 3:
            displayArray(arr, size);
            break;
          case 4:
            printf("Exiting the program.\n");
            break;
          default:
            printf("Invalid choice.\n");
        }
    } while (choice != 4);

    return 0;     }
```

Q. 2: Write a program to print the biggest and smallest elements in an array.

```
#include <stdio.h>


void findMinMax(int arr[], int size) {
    if (size <= 0) {
        printf("Array is empty.\n");
        return;
    }


    int min = arr[0], max = arr[0];


    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
        if (arr[i] > max) {
            max = arr[i];
        }
    }


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


int main() {
    int arr[] = {12, 56, 7, 32, 91, 5};
    int size = sizeof(arr) / sizeof(arr[0]);


    findMinMax(arr, size);
    return 0;
}
```

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

```
#include <stdio.h>

void findSumAndAverage(int arr[], int size) {
    if (size <= 0) {
        printf("Array is empty.\n");
        return;
    }

    int sum = 0;
    float average;

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

    average = (float)sum / size;

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

int main() {
    int arr[] = {12, 56, 7, 32, 91, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    findSumAndAverage(arr, size);

    return 0;
}
```

Q. 4: Write a program to sort an array using bubble sort.

```
#include <stdio.h>
void bubbleSort(int arr[], int size) {
    int temp;
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap if the current element is greater than the next element
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

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

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int size = sizeof(arr) / sizeof(arr[0]);

    bubbleSort(arr, size);
    printArray(arr, size);

    return 0;
}
```

Q. 5: Write a program to search for an element using linear search and binary search.

```
#include <stdio.h>

int linearSearch(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return i; // Return the index if element is found
        }
    }
    return -1; // Return -1 if element is not found
}

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

    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] == key) {
            return mid; // Return the index if element is found
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1;
}

int main() {
    int arr[] = {2, 3, 4, 10, 40};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 10;

    int linearResult = linearSearch(arr, size, key);
    if (linearResult != -1) {
        printf("Element found at index (Linear Search): %d\n", linearResult);
    } else {
        printf("Element not found (Linear Search).\n");
    }

    int binaryResult = binarySearch(arr, size, key);
    if (binaryResult != -1) {
        printf("Element found at index (Binary Search): %d\n", binaryResult);
    } else {
        printf("Element not found (Binary Search).\n");
    }

    return 0;
```

Q. 6: Take an array of 20 integer inputs from the user and print the counts of positive, negative, odd, even, and zero numbers.

```
#include <stdio.h>


int main() {
   int arr[20];
   int positiveCount = 0, negativeCount = 0, oddCount = 0, evenCount = 0, zeroCount = 0;


   printf("Enter 20 integers:\n");
   for (int i = 0; i < 20; i++) {
      scanf("%d", &arr[i]);


      if (arr[i] > 0) {
         positiveCount++;
      } else if (arr[i] < 0) {
         negativeCount++;
      }


      if (arr[i] % 2 == 0 && arr[i] != 0) {
         evenCount++;
      } else if (arr[i] % 2 != 0) {
         oddCount++;
      }


      if (arr[i] == 0) {
         zeroCount++;
      }
   }
```

```
    printf("Positive numbers: %d\n", positiveCount);
    printf("Negative numbers: %d\n", negativeCount);
    printf("Odd numbers: %d\n", oddCount);
    printf("Even numbers: %d\n", evenCount);
    printf("Zeroes: %d\n", zeroCount);

    return 0;
}
```

Q. 7: Take an array of 10 elements, split it into two different arrays from the middle, and display them.

```
#include <stdio.h>

void splitArray(int arr[], int size) {
    if (size % 2 != 0) {
        printf("The array size should be even for splitting into two halves.\n");
        return;
    }

    int middle = size / 2;
    int firstHalf[middle], secondHalf[middle];

    for (int i = 0; i < middle; i++) {
        firstHalf[i] = arr[i];
        secondHalf[i] = arr[middle + i];
    }

    printf("First Half: ");
    for (int i = 0; i < middle; i++) {
        printf("%d ", firstHalf[i]);
    }
    printf("\nSecond Half: ");
    for (int i = 0; i < middle; i++) {
        printf("%d ", secondHalf[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {58, 24, 13, 15, 63, 9, 8, 81, 1, 78};
    int size = sizeof(arr) / sizeof(arr[0]);

    splitArray(arr, size);

    return 0;
}
```

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

```
#include <stdio.h>
void findFrequency(int arr[], int size) {
    if (size <= 0) {
        printf("Array is empty.\n");
        return;
    }

    int freq[size]; // Array to store frequency of each element
    int visited = -1; // Mark all elements as visited
    for (int i = 0; i < size; i++) {
        freq[i] = 0;
    }
    for (int i = 0; i < size; i++) {
        int count = 1;
        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                count++;
                freq[j] = visited;
            }
        }
        if (freq[i] != visited) {
            freq[i] = count;
        }
    }

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

int main() {
    int arr[] = {3, 4, 2, 5, 6, 3, 5, 3, 6, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);

    findFrequency(arr, size);
    return 0;
}
```