Q1>Find the sum of all the multiples of 3 or 5 below 1000.

```
#include <stdio.h>

int main() {
   int sum = 0;

       for (int i = 0; i < 1000; i++) {
             if (i % 3 == 0 || i % 5 == 0) {
           sum += i;
       }
   }

      printf("The sum of all multiples of 3 or 5 below 1000 is: %d\n", sum);

   return 0;
}
```

Q2>Find all the roots of a quadratic equation ax2+bx+c=0.

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

int main() {
   float a, b, c, discriminant, root1, root2;

      printf("Enter coefficients a, b, and c: ");
      scanf("%f %f %f", &a, &b, &c);

   discriminant = b * b - 4 * a * c;

      if (discriminant >= 0) {

      root1 = (-b + sqrt(discriminant)) / (2 * a);
       root2 = (-b - sqrt(discriminant)) / (2 * a);

             printf("Root 1 = %.2f\n", root1);
       printf("Root 2 = %.2f\n", root2);

   } else {

            printf("No real roots.\n");
   }

   return 0;
}
```

Q3>Find the largest among three different numbers entered by user.

```
#include <stdio.h>

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

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

   if (num1 > num2 && num1 > num3) {
       printf("The largest number is: %d\n", num1);
   } else if (num2 > num1 && num2 > num3) {
       printf("The largest number is: %d\n", num2);
   } else {
       printf("The largest number is: %d\n", num3);
   }

   return 0;
}
```

Q4>Determine and Output Whether Number N is Even or Odd.

```
#include <stdio.h>

int main() {
    int N;

      printf("Enter an integer: ");
    scanf("%d", &N);


    if (N % 2 == 0) {
        printf("%d is an even number.\n", N);
    } else {
        printf("%d is an odd number.\n", N);
    }

    return 0;
}
```

Q5>Calculate the remainder, and the quotient of a given numbers.

```
#include <stdio.h>

int main() {
    int dividend, divisor, quotient, remainder;

    printf("Enter the dividend: ");
    scanf("%d", &dividend);
    printf("Enter the divisor: ");
    scanf("%d", &divisor);

    quotient = dividend / divisor;
    remainder = dividend % divisor;


    printf("Quotient: %d\n", quotient);
    printf("Remainder: %d\n", remainder);

    return 0;
}
```

Q6>Generate even numbers between 100 and 200.

```
#include <stdio.h>

int main() {
    int i;

    printf("Even numbers between 100 and 200 are:\n");
    for (i = 100; i <= 200; i++) {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }

    return 0;
}
```

Lab 2

Q1>Write iterative and recursive code of linear search. Perform time (best, worst, and average) and space complexity analysis of the code. In case of recursive approach, first write recurrence relation then solve it to get the time complexity

```
#include <stdio.h>

int main() {
    int arr[] = {2, 3, 5, 6, 7, 9};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key, i, found = 0;


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


    for (i = 0; i < size; i++) {
        if (arr[i] == key) {
            printf("Element found at index: %d\n", i);
            found ++;
            break;
       }
    }


    if (found==0) {
        printf("Element not found\n");
    }

    return 0;
}
```

Q2>Write iterative and recursive code of Binary search. Perform time (best, worst, and average) and space complexity analysis of the code. In case of recursive approach, first write recurrence relation then solve it to get the time complexity.

```
#include <stdio.h>

int main() {
    int arr[] = {1, 3, 5, 7, 9, 11, 13, 15};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key, low = 0, high = size - 1, mid;


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


    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] == key) {
            printf("Element found at index: %d\n", mid);
            return 0;
        }
        else if (arr[mid] < key) {
            low = mid + 1;
        }
        else {
            high = mid - 1;
        }
    }

    printf("Element not found\n");
    return 0;
}
```

Lab 3

Q1>Write function to calculate the Sum of n natural numbers using recursion and perform time and space complexity analysis of the function.

```
#include <stdio.h>


int sumOfNaturalNumbers(int n) {

    if (n == 0) {
        return 0;
    }

    return n + sumOfNaturalNumbers(n - 1);
}

int main() {
    int n;


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


    int result = sumOfNaturalNumbers(n);
    printf("The sum of the first %d natural numbers is: %d\n", n, result);

    return 0;
}
```

Q2>Write a program to print first n Fibonacci numbers using recursion. Write recurrence relation for the function and perform time and space complexity analysis.

```
#include <stdio.h>

int main() {
    int n;

    printf("Enter the number of Fibonacci numbers to print: ");
    scanf("%d", &n);

      int i = 0;
    while (i < n) {
        if (i == 0) {
            printf("0 ");
       } else if (i == 1) {
            printf("1 ");
        } else {
            int a = 0, b = 1, next;
            for (int j = 2; j <= i; j++) {
                next = a + b;
                a = b;
                b = next;               }
            printf("%d ", next);
       }
        i++;
    }

    printf("\n");
    return 0;
}
```

Q3>Write a program for binary search using recursion. Write recurrence relation for the function and perform time and space complexity analysis.

```
#include <stdio.h>

int main() {
    int arr[] = {2, 3, 4, 20, 30, 40};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key, low = 0, high = size - 1, mid;


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


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

        if (arr[mid] == key) {  // Base case: key found
            printf("Element found at index: %d\n", mid);
            return 0;
        }


        if (arr[mid] > key) {
            high = mid - 1;  // Search in the left subarray
        } else {
            low = mid + 1;   // Search in the right subarray
        }
    }


    printf("Element not found\n");
    return 0;
}
```

Q4>Write a program to find the factorial of a number using recursion. Write recurrence relation for the function and perform time and space complexity analysis.

```
#include <stdio.h>

int main() {
    int n;


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


    if (n < 0) {
        printf("negative numbers.\n");
    } else {
        int result = 1;


        int i = n;
        while (i > 1) {
            result *= i;
            i--;
        }

        printf("Factorial of %d is: %d\n", n, result);
    }

    return 0;
}
```

Q5>Write a program to delete a linked list using recursion. Write recurrence relation for the function and perform time and space complexity analysis.

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

struct Node {
    int data;
    struct Node* next;
};

void deleteList(struct Node** head) {
    if (*head == NULL) return;
    deleteList(&(*head)->next);
    free(*head);
    *head = NULL;
}

int main() {
    struct Node* head = malloc(sizeof(struct Node));
    head->data = 1;
    head->next = malloc(sizeof(struct Node));
    head->next->data = 2;
    head->next->next = NULL;

    deleteList(&head);
    if (head == NULL) {
        printf("Linked list deleted successfully.\n");
    }
    return 0;
}
```

Lab 4

Q1>Selection sort.

```
#include <stdio.h>

int main() {
    int arr[] = {11, 12, 22 , 25, 64};
    int n = sizeof(arr) / sizeof(arr[0]);
   int i, j, minindex, temp;


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


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


        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minindex]) {
                minindex = j;
            }
        }

        temp = arr[minindex];
        arr[minindex] = arr[i];
        arr[i] = temp;
    }


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

    return 0;
}
```

Q2>Bubble sort.

```
#include <stdio.h>

int main() {
    int arr[] = {10, 14, 15, 12, 22};
    int n = sizeof(arr) / sizeof(arr[0]);
    int i, j;
    int temp;


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


    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {

                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Q3>Insertion sort

```
#include <stdio.h>

int main() {
    int arr[] = {4, 14, 25, 20, 11};
    int n = sizeof(arr) / sizeof(arr[0]);
   int i, j, key;


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


    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;


        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Q4>Merge sort   // Copied from net

```
#include <stdio.h>

int main() {
    int arr[] = {38, 27, 43, 3, 9, 82, 10}; // Sample array
    int n = sizeof(arr) / sizeof(arr[0]); // Calculate number of elements

    // Print original array
    printf("Given array is: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // Merge Sort Algorithm
    for (int size = 1; size < n; size *= 2) { // size of subarrays
        for (int left = 0; left < n; left += 2 * size) { // left index of subarray
            // Find mid and right index
            int mid = left + size - 1;
            int right = left + 2 * size - 1;
            if (mid >= n) break; // If mid exceeds array length
            if (right >= n) right = n - 1; // Adjust right index to stay within
bounds

            // Calculate sizes of the two subarrays to be merged
            int n1 = mid - left + 1; // Size of left subarray
            int n2 = right - mid;    // Size of right subarray
            // Create temporary arrays for merging
            int L[n1], R[n2];

            // Copy data to temporary arrays L[] and R[]
            for (int i = 0; i < n1; i++)
                L[i] = arr[left + i];
            for (int j = 0; j < n2; j++)
                R[j] = arr[mid + 1 + j];

            // Merge the temporary arrays back into arr[left..right]
            int i = 0, j = 0, k = left;
            while (i < n1 && j < n2) {
                if (L[i] <= R[j]) {
                    arr[k++] = L[i++];
                } else {
                    arr[k++] = R[j++];
                }
            }

            // Copy remaining elements of L[] if any
            while (i < n1) {
                arr[k++] = L[i++];
            }

            // Copy remaining elements of R[] if any
            while (j < n2) {
                arr[k++] = R[j++];
            }
        }
    }

    // Print sorted array
    printf("\nSorted array is: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

Lab 5