1.

```c
/*
Exercise 1: Write a program to convert English units to metric (i.e., miles to
kilometers, gallons to liters, etc.). Include a specification and a code
design.
*/
#include <stdio.h>

void displayMenu() {
    printf("\nUnit Conversion Program\n");
    printf("1. Miles to Kilometers\n");
    printf("2. Gallons to Liters\n");
    printf("3. Pounds to Kilograms\n");
    printf("4. Inches to Centimeters\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
}

int main() {
    int choice;
    float value, result;

    do {
        displayMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter miles: ");
                scanf("%f", &value);
                result = value * 1.60934;
                printf("%.2f miles = %.2f kilometers\n", value, result);
                break;
            case 2:
                printf("Enter gallons: ");
                scanf("%f", &value);
                result = value * 3.78541;
                printf("%.2f gallons = %.2f liters\n", value, result);
                break;
            case 3:
                printf("Enter pounds: ");
                scanf("%f", &value);
                result = value * 0.453592;
                printf("%.2f pounds = %.2f kilograms\n", value, result);
                break;
            case 4:
                printf("Enter inches: ");
                scanf("%f", &value);
```

```
                result = value * 2.54;
                printf("%.2f inches = %.2f centimeters\n", value, result);
                break;
            case 5:
                printf("Exiting the program. Goodbye!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 5);

    return 0;
}
```

2.

```
//1. C program to find the HCF (Highest Common Factor) of given numbers using
recursion

#include <stdio.h>

int HCF(int,int);

int main()
{
    int n1,n2,gcd;

    printf("\nEnter the two number's to find the HCF : ");
    scanf("%d %d",&n1,&n2);

    gcd = HCF(n1,n2);
    printf("\nHCF of the given two number's %d & %d is %d",n1,n2,gcd);

    return 0;
}

int HCF(int n1,int n2)
{
    if(n2!= 0)
    {
        return HCF(n2,n1%n2);
    }
    else
        return n1;
}
```

3.

```c
// 2. Cprogram to find the LCM (Lowest Common Multiple) of given numbers using recursion
#include <stdio.h>


int LCM(int,int,int);

int main()
{
    int n1,n2,lcm;

    printf("\nEnter the two number's to find there LCM : ");
    scanf("%d %d",&n1,&n2);


    lcm = LCM(n1,n2,1);

    printf("\nLCM of the given two number's %d and %d is %d.",n1,n2,lcm);
    return 0;
}
int LCM(int n1,int n2,int multiple)
{
    int m = n1 * multiple;
    if(m%n2==0)
    {
        return m;
    }

    return LCM(n1,n2,multiple+1);

}
```

4.

```c
//6. Cprogram to convert a Binary number to Gray Code using Recursion
#include <stdio.h>

int binaryToGrayRecursive(int);

// Function to calculate Gray Code using recursion


int main()
{
    int binary, gray;

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

```
    // Convert binary to Gray Code using recursion
    gray = binaryToGrayRecursive(binary);

    printf("Gray Code equivalent of binary number %d is: %d\n", binary, gray);

    return 0;
}
int binaryToGrayRecursive(int binary)
{
    if (binary == 0)
        return 0; // Base case: Gray Code of 0 is 0
    int higherBits = binary >> 1; // Right shift binary by 1
    return (binary ^ higherBits); // XOR current binary with the shifted bits
}
```

5.

```
//3. C program to find the GCD (Greatest Common Divisor) of given numbers
using recursion
#include <stdio.h>

int GCD(int,int);

int main()
{
    int n1,n2,gcd;

    printf("\nEnter the number's to find the GCD : ");
    scanf("%d %d",&n1,&n2);

    gcd = GCD(n1,n2);

    printf("\nThe GCD of %d and %d is %d.",n1,n2,gcd);

    return 0;
}
int GCD(int n1,int n2)
{
    if(n2!=0)
    {
        return GCD(n2,n1%n2);
    }
    else
        return n1;
}
```

6.

```c
//5. C programs to convert a linary number to Gray Code
#include <stdio.h>

// Function to convert binary to Gray Code
int binaryToGray(int binary)
{
    return binary ^ (binary >> 1); // XOR the binary number with itself
shifted one position to the right
}

int main()
{
    int binary, gray;

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

    gray = binaryToGray(binary);

    printf("Gray Code equivalent of binary number %d is: %d\n", binary, gray);

    return 0;
}
```

7.

```c
/*
7. C program to print following Pyramid:









*/
#include <stdio.h>

int main() {
    int n = 5; // Number of rows

    for (int i = n; i >= 1; i--) {
        // Print leading spaces for the mirror effect
```

```
        // Print stars for the left side of the pyramid
        for (int j = 1; j <= i; j++) {
            printf("* ");
        }
        for (int j = n; j > i; j--) {
            printf("  "); // Two spaces for alignment
        }
        // Print stars for the right side of the pyramid
        for (int j = 1; j <= i; j++) {
            printf("  *");
        }
        printf("\n"); // Move to the next line
    }

    return 0;
}
```

8.

```
//11. C Program to Read a Matrix and Print Diagonals
#include <stdio.h>

void printDiagonals(int matrix[100][100], int n) {
    printf("Main diagonal elements: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", matrix[i][i]); // Elements where row == column
    }
    printf("\n");

    printf("Secondary diagonal elements: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", matrix[i][n - 1 - i]); // Elements where row + column ==
n - 1
    }
    printf("\n");
}

int main() {
    int n, matrix[100][100];

    printf("Enter the size of the square matrix (n x n): ");
    scanf("%d", &n);

    if (n <= 0 || n > 100) {
        printf("Please enter a valid matrix size (1 to 100).\n");
        return 0;
    }
```

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

    printDiagonals(matrix, n);

    return 0;
}
```

9.

```
//4. Cprogram Decimal number to Binary using Recursion.

#include <stdio.h>

void binary(int);

int main()
{
    int n ;
    printf("\nEnter the decimal number : ");
    scanf("%d",&n);
    printf("\nThe Binary value of decimal number %d is : ",n);

    if (n == 0) {
        printf("0"); // Handle case when input is 0
    } else {
        binary(n);
    }



    return 0;
}

void binary(int num)
{
    if(num==0)
        return;

    else {
        binary(num/2);
        printf("%d",num%2);
```

```
    }


}
```

10.

```
/*8. C program to find the sum of Natural Number/Factorial of Number of all
natural numbers from 1 to N.
    Series: 1/1! + 2/2! +3/3!+4/4!+ N/N!*/

#include <stdio.h>

// Function to calculate factorial of a number
int factorial(int num) {
    if (num == 0 || num == 1)
        return 1;
    else
        return num * factorial(num - 1);
}

// Function to calculate the sum of the series
double sumOfSeries(int N) {
    if (N == 1)
        return 1.0; // Base case: 1/1!
    else
        return (N / (double)factorial(N)) + sumOfSeries(N - 1);
}

int main() {
    int N;
    double result;

    printf("Enter the value of N: ");
    scanf("%d", &N);

    if (N <= 0) {
        printf("Please enter a positive integer greater than 0.\n");
        return 0;
    }

    result = sumOfSeries(N);

    printf("The sum of the series up to %d is: %.6f\n", N, result);

    return 0;
}
```

11.

```c
/*
9. C program to find sum of following series
    1+3^2/3^3 + 5^2/5^3 + 7^2/7^3 +... till N terms
*/

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

// Function to calculate the series sum
double sumOfSeries(int N) {
    double sum = 0.0;
    int term = 1; // Start with the first odd number

    for (int i = 1; i <= N; i++) {
        double numerator = pow(term, 2); // Calculate term^2
        double denominator = pow(term, 3); // Calculate term^3
        sum += numerator / denominator; // Add the fraction to the sum
        term += 2; // Move to the next odd number
    }

    return sum;
}

int main() {
    int N;
    double result;

    printf("Enter the number of terms (N): ");
    scanf("%d", &N);

    if (N <= 0) {
        printf("Please enter a positive integer greater than 0.\n");
        return 0;
    }

    result = sumOfSeries(N);

    printf("The sum of the series up to %d terms is: %.6f\n", N, result);

    return 0;
}
```

12.

```c
//10. C program to replace all EVEN elements by 0 and Odd by 1 in One
Dimensional Array
#include <stdio.h>

void replaceEvenOdd(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0)
            arr[i] = 0; // Replace even number with 0
        else
            arr[i] = 1; // Replace odd number with 1
    }
}

int main() {
    int n;

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

    if (n <= 0) {
        printf("Please enter a positive size for the array.\n");
        return 0;
    }

    int arr[n];

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

    replaceEvenOdd(arr, n);

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

    return 0;
}
```

13.

```c
//11. C Program to Read a Matrix and Print Diagonals
#include <stdio.h>
```

```c
void printDiagonals(int matrix[100][100], int n) {
    printf("Main diagonal elements: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", matrix[i][i]); // Elements where row == column
    }
    printf("\n");

    printf("Secondary diagonal elements: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", matrix[i][n - 1 - i]); // Elements where row + column ==
n - 1
    }
    printf("\n");
}

int main() {
    int n, matrix[100][100];

    printf("Enter the size of the square matrix (n x n): ");
    scanf("%d", &n);

    if (n <= 0 || n > 100) {
        printf("Please enter a valid matrix size (1 to 100).\n");
        return 0;
    }

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

    printDiagonals(matrix, n);

    return 0;
}
```

14.

```c
//13. C program to input and print text using Dynamic Memory Allocation.
#include <stdio.h>
#include <stdlib.h> // For malloc() and free()

int main() {
    char *text;
    int size;
```

```
    printf("Enter the number of characters for the text (including spaces):
");
    scanf("%d", &size);

    // Dynamically allocate memory for the text
    text = (char *)malloc((size + 1) * sizeof(char)); // +1 for the null
terminator
    if (text == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    // Clear the input buffer
    getchar(); // Consume the newline left by scanf

    printf("Enter the text: ");
    fgets(text, size + 1, stdin); // Read the text from the user

    printf("The entered text is: %s", text);

    // Free the allocated memory
    free(text);

    return 0;
}
```

15.

```
//14. C. program to read a one dimensional array, print sum of all elements
along with inputted array elements using Dynamic Memory Allocation.

#include <stdio.h>
#include <stdlib.h> // For malloc() and free()

int main() {
    int *arr;
    int n, sum = 0;

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

    // Dynamically allocate memory for the array
    arr = (int *)malloc(n * sizeof(int));
    if (arr == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
```

```
    }

    // Input array elements
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i]; // Calculate the sum
    }

    // Print the array elements
    printf("The elements of the array are: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

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

    // Free the allocated memory
    free(arr);

    return 0;
}
```

16.

```
/*Exercise 2: Write a program to perform date arithmetic such as how many days
there are between 6/6/90 and 4/3/92. Include a specification and a code
design.
*/
#include <stdio.h>

// Function prototypes
int isLeapYear(int year);
int daysInMonth(int month, int year);
int dateToDays(int day, int month, int year);
int calculateDifference(int day1, int month1, int year1, int day2, int month2,
int year2);

int main() {
    int day1, month1, year1;
    int day2, month2, year2;
    int difference;

    // Input the first date
    printf("Enter the first date (DD/MM/YY): ");
```

```
    scanf("%d/%d/%d", &day1, &month1, &year1);

    // Input the second date
    printf("Enter the second date (DD/MM/YY): ");
    scanf("%d/%d/%d", &day2, &month2, &year2);

    // Calculate the difference
    difference = calculateDifference(day1, month1, year1, day2, month2,
year2);

    // Output the result
    printf("The number of days between %d/%d/%d and %d/%d/%d is %d days.\n",
           day1, month1, year1, day2, month2, year2, difference);

    return 0;
}

// Function to check if a year is a leap year
int isLeapYear(int year) {
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        return 1;
    }
    return 0;
}

// Function to get the number of days in a month
int daysInMonth(int month, int year) {
    switch (month) {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            return 31;
        case 4: case 6: case 9: case 11:
            return 30;
        case 2:
            return isLeapYear(year) ? 29 : 28;
        default:
            return 0; // Invalid month
    }
}

// Function to convert a date to the total number of days since 01/01/0000
int dateToDays(int day, int month, int year) {
    int totalDays = 0;

    // Add days for the years before the given year
    for (int i = 0; i < year; i++) {
        totalDays += isLeapYear(i) ? 366 : 365;
    }
```

```
    // Add days for the months before the given month in the current year
    for (int i = 1; i < month; i++) {
        totalDays += daysInMonth(i, year);
    }

    // Add the days in the current month
    totalDays += day;

    return totalDays;
}

// Function to calculate the difference between two dates
int calculateDifference(int day1, int month1, int year1, int day2, int month2,
int year2) {
    int days1 = dateToDays(day1, month1, year1);
    int days2 = dateToDays(day2, month2, year2);

    return days2 - days1;
}
```

17.

```
/*
Exercise 3: A serial transmission line can transmit 960 characters each
second.
Write a program that will calculate the time required to send a file, given
the file's
size. Try the prog ram on a 400MB (419,430,400 -byte) file. Use appropriate
units.
(A 400MB file takes days.)

*/
#include <stdio.h>

void calculateTransmissionTime(long long fileSize, int rate);

int main() {
    long long fileSize;  // File size in bytes
    int rate = 960;      // Transmission rate in characters per second

    // Input file size
    printf("Enter the file size in bytes: ");
    scanf("%lld", &fileSize);

    // Calculate transmission time
    calculateTransmissionTime(fileSize, rate);
```

```
    return 0;
}

void calculateTransmissionTime(long long fileSize, int rate) {
    double timeInSeconds = (double)fileSize / rate;  // Time in seconds

    // Display time in appropriate units
    if (timeInSeconds < 60) {
        printf("Time required to transmit the file: %.2f seconds\n",
timeInSeconds);
    } else if (timeInSeconds < 3600) {
        printf("Time required to transmit the file: %.2f minutes\n",
timeInSeconds / 60);
    } else if (timeInSeconds < 86400) {
        printf("Time required to transmit the file: %.2f hours\n",
timeInSeconds / 3600);
    } else {
        printf("Time required to transmit the file: %.2f days\n",
timeInSeconds / 86400);
    }
}
```

18.

```
/*
Exercise 4: Write a program to add an 8% sales tax to a given amount and round
the result to the nearest penny.


*/
#include <stdio.h>
#include <math.h>

double calculateTotalWithTax(double amount);

int main() {
    double amount, total;

    // Input the original amount
    printf("Enter the amount in dollars: ");
    scanf("%lf", &amount);

    // Calculate the total with 8% sales tax
    total = calculateTotalWithTax(amount);

    // Display the result
    printf("The total amount after adding 8%% sales tax is: $%.2f\n", total);
```

```
    return 0;
}

double calculateTotalWithTax(double amount) {
    double taxRate = 0.08;                  // Sales tax rate (8%)
    double total = amount + (amount * taxRate); // Total amount with tax
    return round(total * 100) / 100;        // Round to the nearest penny
}
```

19.

```
/*
Exercise 5: Write a program to tell if a number is prime.
*/
#include <stdio.h>
#include <stdbool.h>

bool isPrime(int num);

int main() {
    int num;

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

    // Check if the number is prime
    if (isPrime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}

bool isPrime(int num) {
    // Check for numbers less than 2
    if (num <= 1) {
        return false; // Numbers less than 2 are not prime
    }

    // Check if the number is divisible by any number from 2 to sqrt(num)
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return false; // If divisible, not a prime
```

```
        }
    }

    return true; // If no divisors found, it's a prime number
}
```

20.

```
/*
Exercise 6: Write a program that takes a series of numbers and counts the
number of positive and negative values.

*/
#include <stdio.h>

int main() {
    int num;
    int positiveCount = 0, negativeCount = 0;

    printf("Enter numbers (enter 0 to stop):\n");

    // Continuously input numbers until 0 is entered
    while (1) {
        printf("Enter a number: ");
        scanf("%d", &num);

        // Sentinel value (0) to end the input loop
        if (num == 0) {
            break;
        }

        // Check if the number is positive or negative
        if (num > 0) {
            positiveCount++;
        } else if (num < 0) {
            negativeCount++;
        }
    }

    // Display the results
    printf("Positive numbers: %d\n", positiveCount);
    printf("Negative numbers: %d\n", negativeCount);

    return 0;
}
```

3.

```c
// 2. Cprogram to find the LCM (Lowest Common Multiple) of given numbers using recursion
#include <stdio.h>


int LCM(int,int,int);

int main()
{
    int n1,n2,lcm;

    printf("\nEnter the two number's to find there LCM : ");
    scanf("%d %d",&n1,&n2);


    lcm = LCM(n1,n2,1);

    printf("\nLCM of the given two number's %d and %d is %d.",n1,n2,lcm);
    return 0;
}
int LCM(int n1,int n2,int multiple)
{
    int m = n1 * multiple;
    if(m%n2==0)
    {
        return m;
    }

    return LCM(n1,n2,multiple+1);

}
```