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 unit_conversion() {
int choice;
float value, result;
printf("Choose a conversion type:\n");
printf("1. Miles to Kilometers\n");
printf("2. Gallons to Liters\n");
printf("3. Pounds to Kilograms\n");
printf("Enter your choice: ");
scanf("%d", &choice);
printf("Enter the value to convert: ");
scanf("%f", &value);
switch (choice) {
case 1:
result = value * 1.60934;
printf("%.2f miles = %.2f kilometers\n", value, result);
break;
case 2:
result = value * 3.78541;
printf("%.2f gallons = %.2f liters\n", value, result);
break;
case 3:
result = value * 0.453592;
printf("%.2f pounds = %.2f kilograms\n", value, result);
break;
default:
printf("Invalid choice.\n");
}
}
int main() {
unit_conversion();
return 0;
}
```

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>
#include <time.h>
int days_between_dates(int d1, int m1, int y1, int d2, int m2, int y2) {
struct tm date1 = {0}, date2 = {0};
date1.tm_mday = d1;
date1.tm_mon = m1 - 1;
date1.tm_year = y1 - 1900;
```

```c
date2.tm_mday = d2;
date2.tm_mon = m2 - 1;
date2.tm_year = y2 - 1900;
time_t time1 = mktime(&date1);
time_t time2 = mktime(&date2);
return (int)difftime(time2, time1) / (60 * 60 * 24);
}
int main() {
int d1, m1, y1, d2, m2, y2;
printf("Enter the first date (dd mm yyyy): ");
scanf("%d %d %d", &d1, &m1, &y1);
printf("Enter the second date (dd mm yyyy): ");
scanf("%d %d %d", &d2, &m2, &y2);
int days = days_between_dates(d1, m1, y1, d2, m2, y2);
printf("Number of days between the dates: %d\n", days);
return 0;
}
```

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.)

```c
#include <stdio.h>
void transmission_time(long file_size) {
double bytes_per_second = 960.0;
double seconds = file_size / bytes_per_second;
int days = (int)(seconds / (24 * 3600));
seconds = seconds - (days * 24 * 3600);
int hours = (int)(seconds / 3600);
seconds = seconds - (hours * 3600);
int minutes = (int)(seconds / 60);
seconds = seconds - (minutes * 60);
printf("Time to transmit the file:\n");
printf("%d days, %d hours, %d minutes, %.0f seconds\n", days, hours, minutes,
seconds);
}
int main() {
long file_size = 419430400; // 400MB in bytes
transmission_time(file_size);
return 0;
}
```

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

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

```
void calculate_sales_tax(float amount) {
float tax = amount * 0.08;
float total = amount + tax;
printf("Original Amount: $%.2f\n", amount);
printf("Total Amount with 8%% Tax: $%.2f\n", roundf(total * 100) / 100);
}
int main() {
float amount;
printf("Enter the amount: ");
scanf("%f", &amount);
calculate_sales_tax(amount);
return 0;
}
```

Exercise 5: Write a program to tell if a number is prime.

```
#include <stdio.h>
int is_prime(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 num;
printf("Enter a number: ");
scanf("%d", &num);
if (is_prime(num))
printf("%d is a prime number.\n", num);
else
printf("%d is not a prime number.\n", num);
return 0;
}
```

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

```
#include <stdio.h>
void count_positive_negative() {
int n, num, positives = 0, negatives = 0;
printf("How many numbers will you enter? ");
scanf("%d", &n);
printf("Enter the numbers:\n");
for (int i = 0; i < n; i++) {
scanf("%d", &num);
if (num > 0)
positives++;
else if (num < 0)
negatives++;
}
printf("Positive numbers: %d\n", positives);
printf("Negative numbers: %d\n", negatives);
}
int main() {
count_positive_negative();
return 0;
}
/*C program to find hcf of a given number using recursion*/
#include<stdio.h>
int hcf(int a,int b);
int main() {
int num1,num2;
printf("enter 2 numbers:");
scanf("%d%d",&num1,&num2);
printf("hcf of %d and %d is: %d\n", num1,num2,hcf(num1,num2));
return 0;
}
int hcf(int a,int b) {
if(b==0){
return a;
}
return hcf(b,a%b);
}
/*C program to find lcm of a given number using recursion*/
#include<stdio.h>
int hcf(int a,int b);
int lcm(int a,int b);
int main() {
int num1,num2;
printf("enter 2 numbers:");
scanf("%d%d",&num1,&num2);
printf("lcm of %d and %d is: %d\n", num1,num2,lcm(num1,num2));
return 0;
}
int hcf(int a,int b) {
if(b==0){
return a;
}
return hcf(b,a%b);
```

```
}
int lcm(int a,int b) {
return a*b/hcf(a,b);
}
/*C program to find gcd of a given number using recursion*/
#include<stdio.h>
int gcd(int a,int b);
int main() {
int num1,num2;
printf("enter 2 numbers:");
scanf("%d%d",&num1,&num2);
printf("gcd of %d and %d is: %d\n", num1,num2,gcd(num1,num2));
return 0;
}
int gcd(int a,int b) {
if(b==0){
return a;
}
return gcd(b,a%b);
}
/C program to convert a decimal number to binary using recursion./
#include <stdio.h>
//function prototype
void decimalToBinary(int decimalNumber);
int main() {
int decimalNumber;
// Get user input
printf("Enter a decimal number: ");
scanf("%d", &decimalNumber);
// Special case for zero
if (decimalNumber == 0) {
printf("Binary representation of %d is: 0\n", decimalNumber);
} else {
printf("Binary representation of %d is: ", decimalNumber);
decimalToBinary(decimalNumber); // Convert to binary and print
printf("\n");
}
return 0;
}
// Recursive function to print binary representation
void decimalToBinary(int decimalNumber) {
if (decimalNumber > 1) {
decimalToBinary(decimalNumber / 2); // Recursively divide the number by 2
}
```

```
printf("%d", decimalNumber % 2); // Print the remainder (binary digit)
}
/*c program to convert binary to grey code*/
#include <stdio.h>
#include <math.h>
// Function to convert binary number to Gray code
void binaryToGray(int num) {
int gray = num ^ (num >> 1); // XOR with the right-shifted number
printf("Gray Code: ");
int n = (int)log2(num) + 1; // Calculate the number of bits in the binary number
for (int i = n - 1; i >= 0; i--) {
printf("%d", (gray >> i) & 1); // Print each bit of Gray code
}
printf("\n");
}
int main() {
int num;
printf("Enter a binary number: ");
scanf("%d", &num);
binaryToGray(num); // Call function to convert binary to Gray code
return 0;
}
/*c program to convert binary to grey code using recursion*/
#include <stdio.h>
// Function to calculate and print the Gray code recursively
void binaryToGrayRecursive(int num, int prevBit) {
// Base case: If the number becomes 0, return
if (num == 0) {
return;
}
// Recursive call with the quotient
binaryToGrayRecursive(num / 2, num % 2);
// Print the current Gray code bit
printf("%d", (num % 2) ^ prevBit); // XOR the current bit with previous bit
}
int main() {
int num;
printf("Enter a decimal number: ");
scanf("%d", &num);
// Print Gray code recursively
printf("Gray Code: ");
binaryToGrayRecursive(num, 0); // Start with 0 as there is no previous bit at the
beginning
printf("\n");
```

```
return 0;
}
/*c program to find the sum of natural /factorial of number of all natural numbers
from 1
to numbers
n series=1/1!+2/2!+3/3!+..........+n/n!*/
#include <stdio.h>
// Function to calculate the factorial of a number
int factorial(int num) {
int fact = 1;
for (int i = 1; i <= num; i++) {
fact *= i;
}
return fact;
}
int main() {
int n;
float sum = 0.0;
// Input the value of n
printf("Enter a number n: ");
scanf("%d", &n);
// Calculating the sum of the series
for (int i = 1; i <= n; i++) {
sum += (float)i / factorial(i);
}
// Output the result
printf("The sum of the series is: %.6f\n", sum);
return 0;
}
/*c program to find the sum of the following series
1+3^2/3^3+52/5^3+7^2/7^3+.............till N terms*/
#include <stdio.h>
// Function to calculate the sum of the series
float sumSeries(int N) {
float sum = 0.0;
// Iterate over the first N terms
for (int i = 1; i <= N; i++) {
int oddNumber = 2 * i - 1; // Calculate the ith odd number
sum += 1.0 / oddNumber; // Add the term 1/oddNumber to the sum
}
return sum;
}
int main() {
int N;
```

```
// Input the number of terms N
printf("Enter the number of terms: ");
scanf("%d", &N);
// Calculate the sum of the series
float result = sumSeries(N);
// Output the result
printf("The sum of the series is: %.6f\n", result);
return 0;
}
/*c program to replace all even numbers by 0 and odd by 1 in one dimensional array*/
#include <stdio.h>
int main() {
int n;
// Input the size of the array
printf("Enter the number of elements: ");
scanf("%d", &n);
int arr[n];
// Input elements of the array
printf("Enter the elements of the array:\n");
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
// Replace even numbers by 0 and odd numbers by 1
for (int i = 0; i < n; i++) {
if (arr[i] % 2 == 0) {
arr[i] = 0; // Even number
} else {
arr[i] = 1; // Odd number
}
}
// Output the modified array
printf("Modified array: ");
for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
printf("\n");
return 0;
}
/*c program to read a matrix and print diagonals*/
#include <stdio.h>
int main() {
int rows, cols;
// Input the number of rows and columns for the matrix
printf("Enter the number of rows and columns: ");
```

```
scanf("%d %d", &rows, &cols);
int matrix[rows][cols];
// Input the elements of the matrix
printf("Enter the elements of the matrix:\n");
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
scanf("%d", &matrix[i][j]);
}
}
// Check if the matrix is square (diagonals exist only in square matrices)
if (rows == cols) {
printf("Primary Diagonal: ");
for (int i = 0; i < rows; i++) {
printf("%d ", matrix[i][i]); // Primary diagonal elements
}
printf("\n");
printf("Secondary Diagonal: ");
for (int i = 0; i < rows; i++) {
printf("%d ", matrix[i][rows - 1 - i]); // Secondary diagonal elements
}
printf("\n");
} else {
printf("The matrix is not square, diagonals do not exist.\n");
}
return 0;
}
/*c program to print upper triangular portion of a 3 *3 matrix
*/
#include <stdio.h>
int main() {
int matrix[3][3];
printf("Enter the elements of the 3x3 matrix:\n");
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
scanf("%d", &matrix[i][j]);
}
}
printf("Upper triangular portion of the matrix:\n");
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
if (j < 3 - i) {
printf("%d ", matrix[i][j]);
} else {
printf(" ");
}
}
printf("\n");
}
return 0;
}
/*c program to input and print text using dynamic memory allocation
*/
#include <stdio.h>
#include <stdlib.h>
int main() {
char *text;
int size;
printf("Enter the size of the text: ");
scanf("%d", &size);
text = (char *)malloc((size + 1) * sizeof(char)); // Allocate memory for the text
if (text == NULL) {
printf("Memory allocation failed!\n");
return 1;
}
printf("Enter the text: ");
getchar(); // Consume the newline left by scanf
fgets(text, size + 1, stdin); // Read the text
printf("You entered: %s", text);
free(text); // Free the allocated memory
return 0;
}
#include <stdio.h>
int main()
{
int n;
printf("Enter the number of rows: ");
scanf("%d", &n);
for(int i = 0; i < n; i++)
{
for(int j = 0; j < n - i; j++)
{
printf("* ");
}
for(int j = 0; j < 2 * i ; j++)
{
printf(" ");
}
for(int j = 0; j < n - i; j++)
```

```
{
printf("* ");
}
printf("\n");
}
return 0;
}
/*c program to read a 1 d array, print sum of all elements along with
inputted array elements using dynamic memory allocation*/
#include <stdio.h>
#include <stdlib.h>
int main() {
int *arr;
int size, sum = 0;
printf("Enter the size of the array: ");
scanf("%d", &size);
arr = (int *)malloc(size * sizeof(int)); // Allocate memory dynamically for the array
if (arr == NULL) {
printf("Memory allocation failed!\n");
return 1;
}
printf("Enter the elements of the array:\n");
for (int i = 0; i < size; i++) {
scanf("%d", &arr[i]);
sum += arr[i]; // Calculate the sum of elements
}
printf("The array elements are: ");
for (int i = 0; i < size; i++) {
printf("%d ", arr[i]);
}
printf("\nSum of the elements: %d\n", sum);
free(arr); // Free the allocated memory
return 0;
}
```