4.1 Write a program for an automatic teller machine that dispenses money. The user should enter the amount desired (a multiple of 10 dollars) and the machine dispenses this amount using the least number of bills. The bills dispensed are 50s, 20s, and 10s. Write a function that determines how many of each kind of bill to dispense.

4.1 Automatic Teller Machine Program

Implementation:

```
#include <stdio.h>

void dispense_money(int amount, int *fifty, int *twenty, int *ten) {
    *fifty = amount / 50;
    amount %= 50;
    *twenty = amount / 20;
    amount %= 20;
    *ten = amount / 10;
}

int main() {
    int amount, fifty, twenty, ten;
    printf("Enter the amount desired (multiple of 10): ");
    scanf("%d", &amount);
    if (amount % 10 != 0) {
        printf("The amount must be a multiple of 10.\n");
        return 1;
    }
    dispense_money(amount, &fifty, &twenty, &ten);
    printf("Bills dispensed:\n");
    printf("$50 bills: %d\n", fifty);
    printf("$20 bills: %d\n", twenty);
    printf("$10 bills: %d\n", ten);
    return 0;
}
```

4.2 Determine the following information about each value in a list of positive integers. a. Is the value a multiple of 7, 11, or 13? b. Is the sum of the digits odd or even? c. Is the value a prime number? You should write a function with three type int output parameters that send back the answers to these three questions. Some sample input data might be: 104 3773 13 121 77 30751

Problem: Analyzing a List of Positive Integers

Implementation:

```
#include <stdio.h>
#include <stdbool.h>
void analyze_number(int num, int *multiple, int *sum_even, int *is_prime) {
    *multiple = (num % 7 == 0) || (num % 11 == 0) || (num % 13 == 0);
    int sum = 0, n = num;
    while (n > 0) {
        sum += n % 10;
        n /= 10;
    }
    *sum_even = (sum % 2 == 0);
    *is_prime = 1;
    if (num <= 1) *is_prime = 0;
    else {
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                *is_prime = 0;
                break;
            }
        }
    }
}

int main() {
    int nums[] = {104, 3773, 13, 121, 77, 30751};
```

```
    int size = sizeof(nums) / sizeof(nums[0]);
    for (int i = 0; i < size; i++) {
        int multiple, sum_even, is_prime;
        analyze_number(nums[i], &multiple, &sum_even, &is_prime);
        printf("Number: %d\n", nums[i]);
        printf("Multiple of 7, 11, or 13: %s\n", multiple ? "Yes" : "No");
        printf("Sum of digits even: %s\n", sum_even ? "Yes" : "No");
        printf("Prime: %s\n", is_prime ? "Yes" : "No");
        printf("\n");
    }
    return 0;
}
```

4.3 The square root of a number N can be approximated by repeated calculation using the formula NG=0.5(LG+N/LG) where NG stands for next guess and LG stands for last guess. Write a function that calculates the square root of a number using this method. i The initial guess will be the starting value of LG . The program will compute a value for NG using the formula given. The difference between NG and LG is checked to see whether these two guesses are almost identical. If they are, NG is accepted as the square root; otherwise, the next guess (NG) becomes the last guess (LG) and the process is repeated (another value is computed for NG, the difference is checked, and so on). The loop should be repeated until the difference is less than 0.005. Use an initial guess of 1.0.

Problem: Square Root Approximation

Implementation:

```
#include <stdio.h>
#include <math.h>
double calculate_sqrt(double N) {
    double LG = 1.0;
    double NG = 0.5 * (LG + N / LG);
    while (fabs(NG - LG) >= 0.005) {
        LG = NG;
        NG = 0.5 * (LG + N / LG);
    }
```

```
    return NG;
}
int main() {
    double N;
    printf("Enter a number to find its square root: ");
    scanf("%lf", &N);
    double sqrt_value = calculate_sqrt(N);
    printf("The approximated square root of %.4f is %.4f\n", N, sqrt_value);
    return 0;
}
```

4.4 When an aircraft or an automobile is moving through the atmosphere, it must overcome a force called drag that works against the motion of the vehicle. The drag force can be expressed as F = 1 2CD× A× ρ× V2 where F is the force (in newtons), CD is the drag coefficient, A is the projected area of the vehicle perpendicular to the velocity vector (in m2), is the density of the gas or fluid through which the body is traveling (kg/m3), and V is the body's velocity. The drag coefficient CD has a complex derivation and is frequently an empirical quantity. Sometimes the drag coefficient has its own dependencies on velocities: For an automobile, the range is from approximately 0.2 (for a very streamlined vehicle) through about 0.5. For simplicity, assume a streamlined passenger vehicle is moving through air at sea level (where ρ = 1.23kg/m3). Write a program that allows a user to input A and CD interactively and calls a function to compute and return the drag force. Your program should call the drag force function repeatedly and display a table showing the drag force for the input shape for a range of velocities from 0 m/s to 40 m/s in increments of 5 m/s.

Problem: Drag Force Calculation

Implementation:

```
#include <stdio.h>
#include <math.h>
#define DENSITY 1.23
double calculate_drag_force(double CD, double A, double V) {
    return 0.5 * CD * A * DENSITY * V * V;
}
int main() {
    double A, CD;
    printf("Enter the projected area of the vehicle (in m^2): ");
```

```
    scanf("%lf", &A);
    printf("Enter the drag coefficient (CD): ");
    scanf("%lf", &CD);
    printf("\nVelocity (m/s)\tDrag Force (N)\n");
    printf("----------------------------\n");
    for (int V = 0; V <= 40; V += 5) {
        double F = calculate_drag_force(CD, A, (double)V);
        printf("%d\t\t%.2f\n", V, F);
    }
    return 0;
}
```