```
/*

   Please write your code in the designated lines below.

*/


/* Add necessary libraries here. */
#include <stdio.h>
#include <math.h>
#define SUM_ODD 101
#define SUM_EVEN 100
/* Add function prototypes here. */
void ATM(int dollars, int *fifties, int *twenties, int *tens);
void test7_11_13(int number, int *IsDivisibleBy7_11_13, int *SumOddOrEven, int
*IsPrimeNumber);
double my_sqrt(double Number);
/* Add main function here. The function calls from the problems below should be
   called in main.
*/
int main()
{
    //ATM
    printf("\n ATM");
    int AmountToDispense = 0;
    int TenDollarBills = 0, TwentyDollarBills = 0, FiftyDollarBills = 0;
    printf("\n Please enter the amount of money to be dispensed (Must be multiple
of 10): ");
    scanf("%d", &AmountToDispense);
    ATM(AmountToDispense, &FiftyDollarBills, &TwentyDollarBills, &TenDollarBills);
    printf("\n Amount requested: %d. Fifties: %d. Twenties: %d. Tens: %d. \n",
AmountToDispense, FiftyDollarBills, TwentyDollarBills, TenDollarBills);

    //Problem 2
    printf("\n Problem Two");
    int UserNumber = 0;
    int IsMultiple = 0, IsSumOddOrEven = 0, IsPrime = 0;
    printf("\n Enter an integer number: ");
    scanf("%d", &UserNumber);
    test7_11_13(UserNumber, &IsMultiple, &IsSumOddOrEven, &IsPrime);
    if(IsMultiple == 1)
    {
            printf("\n The number %d is a multiple of 7, 11, or 13. \n",
UserNumber);
    }
    else
    {
        printf("\n The number %d is not a multiple of 7, 11, 13. \n", UserNumber);
    }

    if(IsSumOddOrEven == SUM_EVEN)
    {
        printf("\n The sum of the digits of number %d: is even. \n", UserNumber);
    }
    else
    {
        printf("\n The sum of the digits of number %d: is odd. \n", UserNumber);
    }
    if(IsPrime == 0)
    {
        printf("\n The number %d: is not a prime number. \n", UserNumber);
    }
    else
    {
        printf("\n The number %d: is a prime number. \n ", UserNumber);
    }

    //Square Root Problem
    printf("\n Square Root Problem");
    double SquareRootNumber = 0.0;
    printf("\n Enter a number: ");
    scanf("%lf", &SquareRootNumber);
    printf("The square root of: %lf is equal to %lf. \n", SquareRootNumber,
my_sqrt(SquareRootNumber));

    return 0;
}


/*
   Add function for problem 1 here.

   1. Write a function 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. The function determines how many of each kind of bill to
   dispense.  The function should be named atm, accept an integer named dollars,
   integer pointer fifties, integer pointer twenties, integer pointer tens and
return
   nothing.  When the function completes, the pointers should be set with the
number
   of 50s, 20s and tens that are necessary to equal dollars.  Call atm from the
main
   function to test it with a few dollar values.  When atm works properly, comment
out
   the function call and start problem 2.
*/

void ATM(int dollars, int *fifties, int *twenties, int *tens)
{
    //Calculate 50 dollar bills
    if (dollars >= 50)
    {
        *fifties = dollars/50;

        dollars -= (*fifties * 50);
    }

    //Calculate 20 dollar bills
    if (dollars >= 20)
    {
        *twenties =  dollars/20;

        dollars -= (*twenties * 20);
    }
    //Calculate 10 dollar bills
    if (dollars >= 10)
    {
        *tens =  dollars/10;

        dollars -= (*tens * 10);
    }
}


/*
   Add function for problem 2 here.

   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 named test7_11_13 with an integer input parameter
and three type
   int output parameters (aka pointers) that send back the answers to these three
questions. If the
   integer is evenly divisible by 7, 11 or 13, its respective output parameter
should be set to 1,
   otherwise zero. Some sample input data might be:

   104 3773 13 121 77 30751

   Call test7_11_13 from the main function to test it.  When test7_11_13 works
properly, comment out
   the function call and start problem 3.
*/
void test7_11_13(int number, int *IsDivisibleBy7_11_13, int *SumOddOrEven, int
*IsPrimeNumber)
{
    //Check if number is divisible 7, 11, or 13
    *IsDivisibleBy7_11_13 = 0;
    if(number % 7 == 0)
    {
        *IsDivisibleBy7_11_13 = 1;
    }
    else if(number % 11 == 0)
    {
        *IsDivisibleBy7_11_13 = 1;
    }
    else if(number % 13 ==0)
    {
        *IsDivisibleBy7_11_13 = 1;
    }

    //Check if the sum of the digits is odd or even
    int SumOfDigits = 0;
    int NumberToTest = number;

    while (NumberToTest > 0)
    {
        SumOfDigits += NumberToTest % 10;
        NumberToTest /= 10;
    }

    if(SumOfDigits % 2 == 0)
    {
        *SumOddOrEven = SUM_EVEN;
    }
    else
    {
        *SumOddOrEven = SUM_ODD;
    }

    //Check if the number is a prime number
    if (number <= 1)
    {
       *IsPrimeNumber = 0;
    }
    else
    {
        *IsPrimeNumber = 1;
        for (int i = 2; i < number; i++)
        {
           if (number % i == 0)
           {
              *IsPrimeNumber = 0;
              break;
           }
        }
    }

}


/*
   3. Add functions for problem 3 here.

   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.  Write a function
   called my_sqrt that accepts an double N, which is the number to find the square
root
   and returns the square root found.  There are no pointers in this problem.  You
do
   have to compare doubles, which requires an approximate match with an allowable
error.

   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.
   Write a driver function and test your square root function for the numbers
   4, 120.5, 88, 36.01, 10,000, and 0.25.
```

```c
*/
double my_sqrt(double Number)
{
    double NG = 0.0, LG = 1.0;
    while(1)
    {
        NG =  0.5 * (LG + (Number/LG));
        if(fabs(NG - LG) < 0.005)
        {
            break;
        }
        else
        {
            LG = NG;
        }
    }
    return NG;
}
```