PROBLEMS

36. Write an *if* statement that will assign the value 1 to the variable `best` if the integer variable `score` is 90 or greater.
37. Repeat Problem 36 using a conditional expression.
38. Write the code to add 4 to an integer variable, `num`, if a float variable, `amount`, is greater than 5.4.
39. Print the value of the integer `num` if the variable `flag` is true.
40. Write the code to print either `zero` or `not zero` based on the integer variable `num`.
41. If the variable `divisor` is not zero, divide the variable `dividend` by `divisor`, and store the result in `quotient`. If `divisor` is zero, assign it to `quotient`. Then print all three variables. Assume that `dividend` and `divisor` are integers and `quotient` is a double.
42. If the variable `flag` is true, read the integer variables `a` and `b`. Then calculate and print the sum and average of both inputs.
43. Rewrite the following code using one *if* statement:

```
if (aChar == 'E')
    c++;
if (aChar == 'E')
    printf ("Value is E\n");
```

44. Rewrite the following code fragment using one *switch* statement:

```
if (ch == 'E' || ch == 'e')
   countE++;
else if (ch == 'A' || ch == 'a')
   countA++
else if (ch == 'I' || ch == 'i')
   countI++;
else
   print ("Error--Not A, E, or I\a\n");
```

45. Write a code fragment that tests the value of an integer `num1`. If the value is 10, square `num1`. If it is 9, read a new value into `num1`. If it is 2 or 3, multiply `num1` by 99 and print out the result. Implement your code using nested *if* statements, not a *switch.*
46. Rewrite Problem 45 using a *switch* statement.
47. Write a code fragment for the flowchart shown in Figure 5-29. Assume that the variables `x` and `y` are integers and `z` is a float-point number.
48. Write a function called `smallest` that, given three integers, returns the smallest one.
49. Write a function called `day_of_week` that, given an integer between 0 and 6, prints the corresponding day of the week. Assume that the first day of the week (0) is Sunday.
50. Write a function called `month_of_year` that, given an integer between 1 and 12, prints the corresponding month of the year.
51. Write a function called `parkingCharge` that, given the type of vehicle (`'c'` for car, `'b'` for bus, `'t'` for truck) and the hours a vehicle spent in the parking lot, returns the parking charge based on the rates shown below.

car	$2 per hour
bus	$3 per hour
truck	$4 per hour

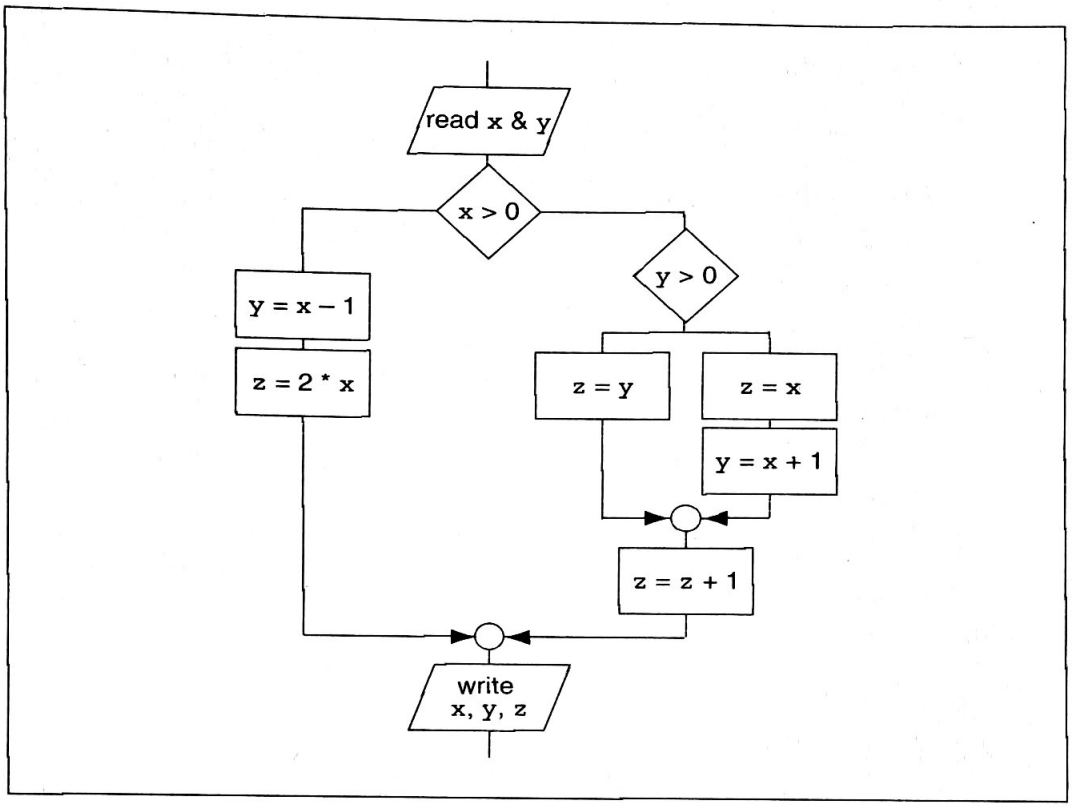

Figure 5-29 Flowchart for Problem 47

52. Write a function to find the smallest of four integers.

PROJECTS

53. Write a program that determines a student's grade. It reads three test scores (between 0 and 100) and calls a function that calculates and returns a student's grade based on the following rules:

    a. If the average score is 90% or more, the grade is `'A'`.

    b. If the average score is 70% or more and less than 90%, it checks the third score. If the third score is more than 90%, the grade is `'A'`, otherwise, the grade is `'B'`.

    c. If the average score is 50% or more and less than 70%, it checks the average of the second and third scores. If the average of the two is greater than 70%, the grade is `'C'`, otherwise, it is `'D'`.

    d. If the average score is less than 50 percent, then the grade is `'F'`.

    The program's `main` is to contain only call statements. At least three sub-functions are required: one to read scores, one to determine the grade, and one to print the results.

54. In Program 4-7, "Strange College Fees," page 134, we wrote a program to calculate college fees. Modify this program for Typical College. At Typical College, the students pay a fee of $10 per unit for up to 12 units; once they have paid for 12 units, they have no additional per-unit fee. The registration fee remains $10 but is assessed only if courses are taken in the term.

55. Given a point, a line from the point forms an angle with the horizontal axis to the right of the line. The line is said to terminate in one of four quadrants based on its angle (`a`) from the horizontal, as shown in Figure 5-30.

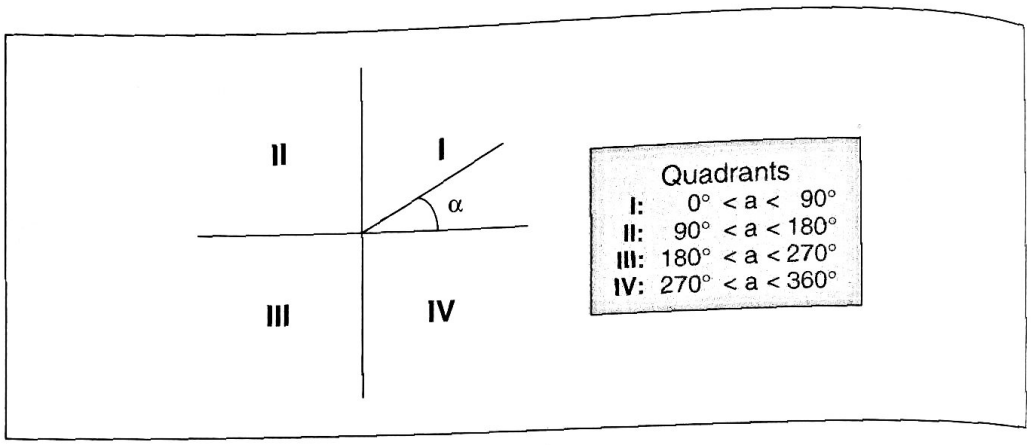

Figure 5-30 Quadrants for Project 55

Write a program that determines the quadrant, given a user-input angle. Use a function to read and validate the angle. Note: If the angle is exactly 0°, it is not in a quadrant but lies on the positive X-axis; if it is exactly 90°, it lies on the positive Y-axis; if it is exactly 180°, it lies on the negative X-axis; and if it is exactly 270°, it lies on the negative Y-axis. Test your program with the following data:

0°, 48.3°, 90°, 179.8°, 180°, 186°, 270°, 300°, and 360°

56. How many values of the variable num must be used to completely test all branches of the following code fragment?

```
if (num > 0)
   if (value < 25)
      {
       value = 10 * num;
       if (num < 12)
           value = value / 10;
      } /* if sum */
   else
      value = 20 * num;
else
   value = 30 * num;
```

57. Write a program that asks the user to enter the current date and a person's birth date in the form month, day, year. The program then calculates the person's age in integral years. Use separate functions to enter the dates (pass by address), calculate the person's age, and print the results. Test your program with the following dates: 11/14/1957, 5/10/1989, and 1/5/2000.

58. Write a C program to calculate the parking fare for customers who park their cars in a parking lot when the following information is given:

a. A character showing the type of vehicle: `'C'` for car, `'B'` for bus, `'T'` for truck.

b. An integer between 0 and 24 showing the hour the vehicle entered the lot.

c. An integer between 0 and 60 showing the minute the vehicle entered the lot.

d. An integer between 0 and 24 showing the hour the vehicle left the lot.

e. An integer between 0 and 60 showing the minute the vehicle left the lot.

This is a public lot. To encourage people to park for a short period of time, the management uses two different rates for each type of vehicle, as shown in Table 5-11.

	First Rate	Second Rate
CAR	$0.00/hr first 3 hr	$1.50/hr after 3 hr
TRUCK	$1.00/hr first 2 hr	$2.30/hr after 2 hr
BUS	$2.00/hr for first hr	$3.70/hr after first hr

Table 5-11 Rates for Project 58

No vehicle is allowed to stay in the parking lot later than midnight; it will be towed away.

The input data consist of a character and a set of four integers representing the type of vehicle and the entering and leaving hours and minutes. But these pieces of data must be input into the computer in a user-friendly way. In other words, the computer must prompt the user to enter each piece of data as shown below. (Bold indicates typical data.)

```
Type of vehicle? C
Hour vehicle entered lot   (0 - 24)?  14
Minute vehicle entered lot (0 - 60)?  23
Hour vehicle left lot      (0 - 24 )? 18
Minute vehicle left lot    (0 - 60)?   8
```

The output format is shown below.

```
                PARKING LOT CHARGE

          Type of vehicle: Car or Bus or Truck
             TIME-IN              XX : XX
             TIME-OUT             XX : XX
                                  --------
             PARKING TIME           XX:XX
             ROUNDED TOTAL             XX
                                  --------
             TOTAL CHARGE           $XX.XX
```

This program must first calculate the actual time spent in the parking lot for each vehicle. This means using modulo arithmetic to handle time calculation. We can calculate this in many ways, one of which is shown below. To calculate the time spent in the parking lot, use the following algorithm:

a. Compare the minute portion of the leaving and the entering time.
If the first one is smaller than the second,
- Add 60 to the minute portion of the leaving time.
- Subtract 1 from the hour portion of the leaving time.

b. Subtract the hour portions.

c. Subtract the minute portions.

d. Since there are no fractional hour charges, the program must also round the parking time up to the next hour before calculating the charge. The program should use the *switch* statement to distinguish between the different types of vehicles.

A well-structured program design is required. A typical solution will use several functions besides `main`. Before you start programming, prepare a structure chart. Run your program six times with the data shown in Table 5-12.

Test	Type	Hour In	Minute In	Hour Out	Minute Out
1	C	12	40	14	22
2	B	8	20	8	40
3	T	2	0	3	59
4	C	12	40	16	22
5	B	8	20	14	20
6	T	2	0	12	0

Table 5-12 Test data for Project 58

59. This program is a simple guessing game. The computer is to generate a random number between 1 and 20. The user is given up to five tries to guess the exact number. After each guess, you are to tell the user if the guessed number is greater than, less than, or equal to the random number. If it is equal, no more guesses should be made. If the user hasn't guessed the number after five tries, display the number with a message that the user should know it by now and terminate the game.

A possible successful dialog:

```
I am thinking of a number between 1 and 20.
    Can you guess what it is?        10
    Your guess is low. Try again:    15
    Your guess is low. Try again:    17
    Your guess is high. Try again:   16

       Congratulations! You did it.
```

A possible unsuccessful dialog:

```
I am thinking of a number between 1 and 20.
    Can you guess what it is?        10
    Your guess is low. Try again:    20
    Your guess is high. Try again:   10
    Your guess is low. Try again:    18
    Your guess is high. Try again:   12

    Sorry. The number was 15.
    You should have gotten it by now.
    Better luck next time.
```

Your design for this program should include a separate function to get the user's guess, a function to print the unsuccessful message, one to print the successful message, and one to print the sorry message.

60. Write a program that, given a person's birth date (or any other date in the Gregorian calendar), will display the day of the week the person was born.

To determine the day of the week, you will first need to calculate the day of the week for December 31 of the previous year. To calculate the day for December 31, use the formula shown below.

$$\left((\text{year} - 1) * 365) + \left\lfloor \frac{(\text{year} - 1)}{4} \right\rfloor - \left\lfloor \frac{\text{year} - 1}{100} \right\rfloor + \left\lfloor \frac{\text{year} - 1}{400} \right\rfloor\right) \% 7$$

The formula determines the day based on the values as shown below.

Day 0: Sunday
Day 1: Monday
Day 2: Tuesday
Day 3: Wednesday
Day 4: Thursday
Day 5: Friday
Day 6: Saturday

Once you know the day for December 31, you simply calculate the days in the year before the month in question. Use a *switch* statement to make this calculation. (Hint: Use *case* 12 first, and then fall into *case* 11, 10 . . . 2.) If the desired month is 12, add the number of days for November (30). If it is 11, add the number of days for October (31). If it is 3, add the number of days for February (28). If it is 2, add the number of days for January (31). If you do not use a *break* between the months, the *switch* will add the days in each month before the current month.

To this figure, add the day in the current month and then add the result to the day code for December 31. This number modulo seven is the day of the week.

There is one more refinement. If the current year is a leap year, and if the desired date is after February, you need to add 1 to the day code. The following formula can be used to determine if the year is a leap year.

```
(!(year % 4) && (year % 100) ) || !(year % 400)
```

Your program should have a function to get data from the user, another to calculate the day of the week, and a third to print the result.

To test your program, run it with the following dates:

a. February 28, 1900, and March 1, 1900

b. February 28, 1955, and March 1, 1955

c. February 28, 1996, and March 1, 1996

d. February 28, 2000, and March 1, 2000

e. December 31, 1996

f. The first and last dates of the current week.

61. Write a program that calculates the change due a customer by denomination; that is, how many pennies, nickels, dimes, etc. are needed in change. The input is to be the purchase price and the size of the bill tendered by the customer ($100, $50, $20, $10, $5, $1).

62. Write a menu-driven program that allows a user to enter five numbers and then choose between finding the smallest, largest, sum, or average. The menu and all the choices are to be functions. Use a *switch* statement to determine what action to take. Provide an error message if an invalid choice is entered.

Run the program five times, once with each option and once with an invalid option. Each run is to use the following set of data:

```
18, 21, 7, 54, 9
```

63. Write a program that tests a user-entered character and displays its classification according to the ASCII classifications shown in Figure 5-25 on page 204. Write the program starting at the top of the classification tree and display all classifications that the character belongs to. For instance, if the user enters a digit, you should display that it is printable, graphical, alphanumeric, and a digit.

64. Write a program to compute the real roots of a quadratic equation ($ax^2 + bx + c = 0$). The roots can be calculated using the following formulas:

$$x1 = -b + \frac{\sqrt{b^2 - 4ac}}{2a}$$

and

$$x2 = -b - \frac{\sqrt{b^2 - 4ac}}{2a}$$

Your program is to prompt the user to enter the constants (`a`, `b`, `c`). It is then to display the roots based on the following rules:

a. If both `a` and `b` are zero, there is no solution.

b. If `a` is zero, there is only one root (`-c / b`).

c. If the discriminate (`b² - 4ac`) is negative, there are no real roots.

d. For all other combinations, there are two roots.

Test your program with the following data:

a	b	c
3	8	5
–6	7	8
0	9	–10
0	0	11