Chapter 1

1. a. true; b. false; c. false; d. true; e. false; f. true; g. true; h. true; i. false; j. false; k. true; l. true; m. true; n. true; o. true; p. false; q. true; r. true; s. true
2. CPU.
3. Base 2 or binary.
4. The equivalent machine language program of a high-level language program.
5. In linking, an object program is combined with other programs in the library, used in the program, to create the executable code.
6. Loader
7. #
8. Preprocessor
9. Programming is a process of problem solving.
10. A step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.
11. (1) Analyze and outline the problem and its solution requirements, and design an algorithm to solve the problem. (2) Implement the algorithm in a programming language, such as C++, and verify that the algorithm works. (3) Maintain the program by using and modifying it if the problem domain changes.
12. (1) Thoroughly understand the problem. (2) Understand the problem requirements. (3) If the problem is complex, divide the problem into subproblems and repeat Steps 1 and 2.
13. To find the weighted average of the four test scores, first you need to know each test score and its weight. Next, you multiply each test score with its weight, and then add these numbers to get the average. Therefore,

    1. Get `testScore1`, `weightTestScore1`
    2. Get `testScore2`, `weightTestScore2`
    3. Get `testScore3`, `weightTestScore3`
    4. Get `testScore4`, `weightTestScore4`

    5.
       ```
       weightedAverage = testScore1 * weightTestScore1 +
                         testScore2 * weightTestScore2 +
                         testScore3 * weightTestScore3 +
                         testScore4 * weightTestScore4;
       ```

14. a. Get `quarters`

    b. Get `dimes`

    c. Get `nickels`

    d. Get `pennies`

    e.
    ```
    changeInPennies = quarters * 25 + dimes * 10 + nickels * 5
                    + pennies
    ```

15. To find the price per square inch, first we need to find the area of the pizza. Then we divide the price of the pizza by the area of the pizza. Let `radius` denote the radius and `area` denote the area of the circle, and `price` denote the price of pizza. Also, let `pricePerSquareInch` denote the price per square inch.

a. Get `radius`

b. `area = π * radius * radius`

c. Get `price`

d. `pricePerSquareInch = price / area`

16. To determine the least selling price of a car, we need to know the listing price of the car. The algorithm is as follows:

    a. Get `listingPrice`

    b. Calculate the least selling price `leastSellingPrice` using the formula:

    `leastSellingPrice = listingPrice × 0.85) + 500`

17. Suppose that `radius` denotes radius of the sphere, `volume` denotes volume of the sphere, and `surfaceArea` denotes the surface area of the sphere. The following algorithm computes the volume and surface area of the sphere.

Algorithm	**C++ Instruction (Code)**
1. Get the radius.	`cin >> radius;`
2. Calculate the volume.	`volume = (4.0 / 3.0) * 3.1416 * radius * radius * radius;`
3. Calculate the surface area.	`surfaceArea = 4.0 * 3.1416 * radius * radius;`

18. Suppose that `billingAmount` denotes the total billing amount, `movingCost` denotes moving cost, `area` denotes the area of the yard that needs to be moved, `numOfAppl` denotes the number of fertilizing applications, and `numOfTrees` denotes the number of trees to be planted. The following algorithm computes and outputs the billing amount.

    a. Enter the area of the yard.

    b. Get `area`

    c. Enter the number of fertilizing applications.

    d. Get `numOfAppl`

    e. Enter the number of trees to be planted.

    f. Get `numOfTrees`

    g. Calculate the `billingAmount` using the formula:

```
billingAmount = (area / 5000) * 35.00 + numOfAppl * 30.00
          + numOfTrees * 50.00
```

19. Suppose that `billingAmount` denotes the total billing amount, `numOfItemsOrdered` denotes the number of items ordered, `shippingAndHandlingFee` denotes the shipping and handling fee, and `price` denotes the price of an item. The following algorithm computes and outputs the billing amount.

```
a. Enter the number of items bought.
b. Get numOfItemsOrdered
c. billingAmount = 0.0;
d. shippingAndHandlingFee = 0.0;
e. Repeat the following for each item bought.
   i. Enter the price of the item
   ii. Get price
   iii. billingAmount = billingAmount + price;
f. if billingAmount < 200
     shippingAndHandlingFee = 10 * numOfItemsOrdered;
g. billingAmount = billingAmount + shippingAndHandlingFee
i. Print billingAmount
```

20. Suppose `amountWithdrawn` denotes the amount to be withdrawn, `serviceCharge` denotes the service charges, if any, and `accountBalance` denotes the total money in the account.

You can now write the algorithm as follows:

```
a.  Get amountWithdrawn.
b.  if amountWithdrawn > 500
      Print "The maximum amount that can be drawn is $500"
    otherwise (if accountBalance <= 0)
       Print "Account balance is <= 0. You cannot withdraw any money."
    otherwise
    {
       if (amountWithdrawn > accountBalance)
       {
            Print "Insufficient balance. If you withdraw, services charges
                   will be $25.00. Select Yes/No."

            if (Yes)
            {
                if (amountWithdrawn > 300)
                    serviceCharge = (amountWithdrawn – 300) * 0.04;
                otherwise
                    serviceCharge = 0;
```

```
                  accountBalance = accountBalance – amountWithdrawn
                                   - serviceCharge – 25;
                  Print "Collect your money. "
              }
          }
      }
      otherwise
      {
           if (amountWithdrawn > 300)
               serviceCharge = (amountWithdrawn – 300) * 0.04;
          otherwise
               serviceCharge = 0;

          accountBalance = accountBalance – amountWithdrawn
                           - serviceCharge;
          Print "Collect your money."
      }
```

21. Suppose `x1` and `x2` are the real roots of the quadratic equation.

    a. Get `a`

    b. Get `b`

    c. Get `c`

    d.
```
if (b * b – 4 * a * c < 0)
  Print "The equation has no real roots."
Otherwise
{
    temp = b * b – 4 * a * c;
    x1 = (-b + temp) / (2 * a);
    x2 = (-b - temp) / (2 * a);
}
```

22. Suppose that `tuition` denotes the tuition per semester, `semesterUnits` denotes the average semester units, `courseUnits` denotes the number of course units, `semesterWeeks` denotes the number of weeks in a semester, `classDaysPerWeek` denotes the number of days a class meets in a week, `classDaysPerSemester` denotes the total number of days a class meets in a semester, `costPerUnit` denotes the cost of one unit of a course, and `costPerClass` denotes the cost of one class.

    a. Get `tuition`

    b. Get `semesterUnits`

    c. Get `semesterWeeks`

    d. Get `courseUnits`

    e. Get `classDaysPerWeek`

    f. Compute `costPerUnit` using the following formula.

```
costPerUnit = tuition / semesterUnits;
```

    g. Compute `classDaysPerSemester` using the following formula.

```
classDaysPerSemester = classDaysPerWeek * semesterWeeks;
```

h. Compute `costPerClass` using the following formula.

```
costPerClass = (costPerUnit * courseUnits) / classDaysPerSemester;
```

23. Suppose `averageTestScore` denotes the average test score, `highestScore` denotes the highest test score, `testScore` denotes a test score, `sum` denotes the sum of all the test scores, `count` denotes the number of students in class, and `studentName` denotes the name of a student.

a. First you design an algorithm to find the average test score. To find the average test score, first you need to count the number of students in the class and add the test score of each student. You then divide the sum by count to find the average test score. The algorithm to find the average test score is as follows:

```
i.   Set sum and count to 0.
ii.  Repeat the following for each student in class.
     1. Get testScore
     2. Increment count and update the value of sum by adding the current test score to sum.
iii. Use the following formula to find the average test score.
     if (count is 0)
         averageTestScore = 0;
     otherwise
         averageTestScore = sum / count;
```

b. The following algorithm determines and prints the names of all the students whose test score is below the average test score.

Repeat the following for each student in class:

```
i.  Get studentName and testScore
ii.
    if (testScore is less than averageTestScore)
        print studentName
```

c. The following algorithm determines the highest test score

```
i.  Get first student's test score and call it highestTestScore.
ii. Repeat the following for each of the remaining students in the class
    1. Get testScore
    2. if (testScore is greater than highestTestScore)
           highestTestScore = testScore;
```

d. To print the names of all the students whose test score is the same as the highest test score, compare the test score of each student with the highest test score and if they are equal print the name. The following algorithm accomplishes this.

Repeat the following for each student in the class:

```
i.  Get studentName and testScore
ii. if (testScore is equal to highestTestScore)
        print studentName
```

You can use the solutions of the subproblems obtained in parts a to d to design the main algorithm as follows:

1. Use the algorithm in part a to find the average test score.
2. Use the algorithm in part b to print the names of all the students whose score is below the average test score.
3. Use the algorithm in part c to find the highest test score.
4. Use the algorithm in part d to print the names of all the students whose test score is the same as the highest test score