Chapter 5  Conditions, Logical Expressions, and Selection Control

Chapter 5

EXERCISE ANSWERS

Exam Preparation Exercises

1. a. T  b. T  c. T  d. F

3. MadamNowI’mAdam

4. a. 10
   10
   b. The value of $x$ is 3
   c. 3
   7
   6
   $z \leftarrow$ Despite the indentation, the final output statement is not part of the else-clause (or even part of the If statement).

5. a. Eligible to serve.
   b. Too short and too light to serve.

7. a. $x < y$ && $y <= z$
    b. $x > 0$ && $y > 0$ && $z > 0$
    c. $x != y$ && $x != z$
    d. $x == y$ && $x == z$

8. a. T  b. F  c. F  d. T  e. T

11. To fix the dangling else problem, insert braces as follows:

    ```
    if (typeA || typeB)
    {
        if (typeA && typeB)
            cout << "Type AB";
    }
    else
        cout << "Type O";
    ```

12. | ch1 ch2 ch3 | Expected Output |
    |-------|----------------|
    | 'A' 'A' 'A' | All initials are the same. |
    | 'A' 'A' 'B' | First two are the same. |
    | 'B' 'A' 'A' | Last two are the same. |
    | 'A' 'B' 'A' | First and last are the same. |
    | 'A' 'B' 'C' | All initials are different. |

14. In the left-to-right evaluation, only the first two relational expressions are evaluated. The first yields true, so evaluation proceeds. The second yields false, so evaluation stops. The final result is false.

Programming Warm-up Exercises

1. ```
   bool eligible;
   eligible = true;
   ```

3. In the following statement, the parentheses are unnecessary but are included for readability.
isCandidate = (satScore >= 1100 && gpa >= 2.5 && age > 15);

5. Solution using a series of If–Then:

   biggest = i;
   if (j > biggest)
       biggest = j;
   if (k > biggest)
       biggest = k;

   Solution using a nested If–Then–Else structure:

   if (i > j)
       if (i > k)
           biggest = i;
       else
           biggest = k;
   else
       if (j > k)
           biggest = j;
       else
           biggest = k;

8. The wrong operator (=) is used in the If condition. The value of the If condition is 4 (interpreted as true), so the then-clause is executed (and, as a side effect of the If test, 50 is stored into length).

10. if (x1 == x2)
    cout << "Slope undefined" << endl;
else
    { 
        m = (y1 - y2) / (x1 - x2);
        cout << "The slope is " << m << endl;
    }

11. discriminant = b * b - 4.0 * a * c;
    if (discriminant < 0.0)
        cout << "No real roots." << endl;
    else
        { 
            root1 = (-b + discriminant) / (2.0 * a);
            root2 = (-b - discriminant) / (2.0 * a);
        }

12. After the call to the open function, insert the following code:

    if ( !info )
        { 
            cout << "Cannot open the input file." << endl;
            return 1;
        }

Case Study Follow–Up

1. The complete test plan for program Notices. The observed output follows each test case.

<table>
<thead>
<tr>
<th>Reason for Test Case</th>
<th>Input Values</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo print check</td>
<td>9999,100, Student Number: 9999 100,100 Test Scores: 100, 100, 100</td>
<td></td>
</tr>
<tr>
<td>Enter a Student ID number and three test scores: Student Number: 9999, Test Scores: 100, 100, 100 Average Score is 100.00--Passing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passing scores</td>
<td>9999,80, Average Score is 80.00 70,90 -- Passing</td>
<td></td>
</tr>
</tbody>
</table>
Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 80, 70, 90
Average Score is 80.00--Passing.

Passing, but marginal scores
9999, 55, 65, 75 -- Passing but marginal

Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 55, 65, 75
Average Score is 65.00--Passing but marginal.

Failing scores
9999, 30, 40, 50 -- Failing

Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 30, 40, 50
Average Score is 40.00--Failing.

Invalid data, Test 1
Invalid Data: Score(s)
9999, -1, 20, 30 less than zero

Invalid data, Test 2
Invalid Data: Score(s)
9999, 10, -1, 30 less than zero

Invalid data, Test 3
Invalid Data: Score(s)
9999, 10, 20, -1 less than zero

Boundary of valid data, Test 1
Average score is 0.00
9999, 0, 0 -- Failing

Boundary of valid data, Test 2
Average score is 10.00
9999, 10, 20 -- Failing

Boundary of valid data, Test 3
Average score is 10.00
9999, 10, 20 -- Failing

Boundary of valid data, Test 4
Average score is 10.00
9999, 10, 20 -- Failing

Boundary of passing but marginal, Test 1
Average score is 60.00
9999, 50, 60, 70 -- Passing but marginal

Boundary of passing but marginal, Test 2
Average score is 69.00
9999, 68, 70
but marginal, Test 2 69, 70 — Passing but marginal
Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 68, 69, 70
Average Score is 69.00—Passing but marginal.

Boundary of failing 9999, 58, Average score is 59.00
59, 60 — Failing
Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 58, 59, 60
Average Score is 59.00—Failing.

Boundary of passing 9999, 60, Average score is 70.00
70, 80 — Passing
Enter a Student ID number and three test scores:
Student Number: 9999, Test Scores: 60, 70, 80
Average Score is 70.00—Passing.

3. Change the present If–Then statement

    if (average < 70.0)
        cout << " but marginal";

to the following If–Then–Else–If:

    if (average < 70.0)
        cout << " but marginal";
    else if (average > 90.0)
        cout << " with high marks";

5. The statement

    if (test1 < 0 || test2 < 0 || test3 < 0)
        dataOK = false;
    else
        dataOK = true;

would be replaced with the following code:

    dataOK = true;
    if (test1 < 0)
        {
            dataOK = false;
            cout << "Invalid Data: Test score 1 is less than zero." << endl;
        }
    if (test2 < 0)
        {
            dataOK = false;
            cout << "Invalid Data: Test score 2 is less than zero." << endl;
        }
    if (test3 < 0)
        {
            dataOK = false;
            cout << "Invalid Data: Test score 3 is less than zero." << endl;
        }

Then, in the “big” If–Then–Else statement

    if (dataOK)
        {
            
    } else
        cout << "Invalid Data: Score(s) less than zero." << endl;
the else part should be deleted.

6. The postcondition for the main module is:
   • the computer has read five integer values into studentID, test1, test2, test3 and test4.
   • the input values have been echo printed
   • the average of the last four input values has been printed if the values are valid
   • either an error message or a message indicating the student’s status has been printed. The message is “Passing,” if the average is \(\geq 70.0\), “Passing but marginal.” if the average is \(< 70.0\) and \(\geq 60.0\), or “Failing.” if the average is \(< 60.0\).

   The postcondition for module Get Data is that it has input five integer values into studentID, test1, test2, test3, and test4.

   The precondition for module Test Data is that test1, test2, test3, and test4 contain meaningful values. Its postcondition is that dataOK contains true if the values in test1, test2, test3, and test4 are non-negative; otherwise, dataOK contains false.

   The precondition for module Calculate Average is that test1, test2, test3 and test4 contain meaningful values. Its postcondition is that the variable average contains the mean of test1, test2, test3, and test4.

   The precondition for module Print Message Indicating Status is that average contains the mean of the values in test1, test2, test3, and test4. Its postcondition is that the value in average has been printed and a message indicating the status of the student has been printed. The message is “Passing,” if average \(\geq 70.0\), “Passing but marginal.” if average \(< 70.0\) and \(\geq 60.0\), or “Failing.” if average \(< 60.0\).

7. A test plan for the modified Notices program that achieves complete code coverage:

<table>
<thead>
<tr>
<th>Reason for Test</th>
<th>Case Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo print check</td>
<td>9999, 90, 90, 90</td>
<td>Student Number: 9999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test Scores: 90, 90, 90</td>
</tr>
<tr>
<td></td>
<td>80, 70, 90</td>
<td>--- Passing</td>
</tr>
<tr>
<td></td>
<td>Average Score is 80.00</td>
<td></td>
</tr>
<tr>
<td>Passing scores</td>
<td>9999, 80, 80</td>
<td>Average score is 65.00</td>
</tr>
<tr>
<td></td>
<td>75, 65, 65</td>
<td>--- Passing but marginal</td>
</tr>
<tr>
<td></td>
<td>Failing scores</td>
<td>Average score is 40.00</td>
</tr>
<tr>
<td></td>
<td>40, 50, 40</td>
<td>--- Failing</td>
</tr>
<tr>
<td>Invalid data, Test 1</td>
<td>9999, -1, 20</td>
<td>Invalid Data: Score(s)</td>
</tr>
<tr>
<td></td>
<td>30, 40</td>
<td>less than zero</td>
</tr>
<tr>
<td>Invalid data, Test 2</td>
<td>9999, 10, -1</td>
<td>Invalid Data: Score(s)</td>
</tr>
<tr>
<td></td>
<td>30, 40</td>
<td>less than zero</td>
</tr>
<tr>
<td>Invalid data, Test 3</td>
<td>9999, 10, 20</td>
<td>Invalid data: Score(s)</td>
</tr>
<tr>
<td></td>
<td>-1, 40</td>
<td>less than zero</td>
</tr>
<tr>
<td>Invalid data, Test 4</td>
<td>9999, 10, 20</td>
<td>Invalid data: Score(s)</td>
</tr>
<tr>
<td></td>
<td>30, -1</td>
<td>less than zero</td>
</tr>
<tr>
<td>Boundary of valid data, Test 1</td>
<td>9999, 0</td>
<td>Average score is 0.00</td>
</tr>
<tr>
<td></td>
<td>0, 0, 0</td>
<td>--- Failing</td>
</tr>
<tr>
<td>Boundary of valid data, Test 2</td>
<td>9999, 0</td>
<td>Average score is 10.00</td>
</tr>
<tr>
<td></td>
<td>10, 20, 10</td>
<td>--- Failing</td>
</tr>
<tr>
<td>Boundary of valid data, Test 3</td>
<td>9999, 10, 0, 20, 10</td>
<td>Average score is 10.00</td>
</tr>
<tr>
<td>Boundary of valid data, Test 4</td>
<td>9999, 10, 20, 0, 10</td>
<td>Average score is 10.00</td>
</tr>
<tr>
<td>Boundary of valid data, Test 5</td>
<td>9999, 10, 20, 10, 0</td>
<td>Average score is 10.00</td>
</tr>
<tr>
<td>Boundary of passing but marginal, Test 1</td>
<td>9999, 50, 60, 60, 70</td>
<td>Average score is 60.00</td>
</tr>
<tr>
<td>Boundary of passing but marginal, Test 2</td>
<td>9999, 68, 69, 70, 69</td>
<td>Average score is 69.00</td>
</tr>
<tr>
<td>Boundary of failing</td>
<td>9999, 58, 59, 60, 59</td>
<td>Average score is 59.00</td>
</tr>
<tr>
<td>Boundary of passing</td>
<td>9999, 60, 70, 80, 70</td>
<td>Average score is 70.00</td>
</tr>
</tbody>
</table>