**4.1 Introduction**

4.1 Which is *not* an iteration statement?

a) continue

b) for

c) while

d) do … while

ANS: (a)

4.2 Which statement is used to skip the remainder of the body of an iteration statement and proceed with the next iteration of the loop?

a) skip

b) proceed

c) continue

d) jump

ANS: (c)

**4.2 Iteration Essentials**

4.3 \_\_\_\_\_\_\_\_ iteration is sometimes called definite iteration.

(a) counter-controlled

(b) sentinel-controlled

(c) variable-controlled

(d) none of these

ANS: (a)

4.4 \_\_\_\_\_\_\_\_ iteration is sometimes called indefinite iteration.

(a) counter-controlled

(b) sentinel-controlled

(c) variable-controlled

(d) none of these

ANS: (b)

4.5 Which statement is *false*?

a) Counter-controlled repetion is sometimes called definite iteration.

b) Sentinel-controlled iteration is sometimes called indefinite iteration.

c) The sentinel value typically indicates “end of data.”

d) In counter-controlled iteration, the control variable is always incremented by 1 each time the group of instructions is performed.

ANS: (d)

4.6 Which statement is *true*?

a) Sentinel values are used to control iteration when the precise number of iterations is known in advance.

b) In a counter controlled-loop, the counter may not count downward.

c) Sentinels must be distinct from regular data items.

d) Sentinel-controlled iteration is often called definite iteration.

ANS: (c)

**4.3 Counter-Controlled Iteration**

4.7 Which of the following does counter-controlled iteration *not* require?

(a) an initial value

(b) a condition that tests for the final value

(c) an increment or decrement by which the control variable is modified each time through the loop

(d) counter controlled iteration requires all of the above

ANS: (d)

4.8 The statement

while (--counter >= 1) {

printf("%s\n", counter % 2 ? "even" : "odd");  
}

can not be rewritten as

(a)   
while (--counter >= 1) {

if (counter % 2) {

puts("even");  
 }

else {

puts("odd");  
 }  
}

(b)   
while (counter >= 1) {

if (counter % 2) {

puts("even");  
 }

else {  
 puts("odd");  
 }  
}

--counter;

(c)while (counter >= 1) {

if (counter % 2) {

puts("even");  
 }

else {

puts("odd");  
 }

--counter;

}

(d)   
do {

printf("%s\n", counter % 2 ? "odd" : "even");

--counter;

} while (counter >= 2);

ANS: (b)

4.9 Which of the following is a poor programming practice?

(a) indenting the statements in the body of each control statement

(b) using floating-point values as the counter in counter-controlled iteration

(c) using more than two levels of nesting

(d) placing vertical spacing above and below control statements

ANS: (b)

4.10 Which is *not* always required by counter-controlled iteration?

a) The name of a control variable (or loop counter).

b) The initial value of the control variable.

c) The decrement by which the control variable is modified each time through the loop.

d) The condition that tests for the final value of the control variable (i.e., whether looping should continue.

ANS: (c)

4.11 In the context of counter-controlled iteration, which of the following is *not* accomplished by the control-variable initialization statement?

int c = 10;

a) Names the control variable.

b) Defines the control variable to be an integer.

c) Specifies the sentinel value.

d) Sets the initial value of the control variable to 10.

ANS: (c)

4.12 Which data type should normally *not* be used to control a counting loop?

a) int

b) float

c) short

d) long

ANS: (b)

4.13 Which statement is generally *false*?

a) Control counting loops with integer values.

b) Indent the statements in the body of each control statement for clarity.

c) Put a blank line before and after each major control statement to make it stand out in the program.

d) The more deeply nested a program is, the easier it is to understand.

ANS: (d)

**4.4 for Iteration Statement**

4.14 Which statement automatically handles all the details of counter-controlled iteration.

a) for

b) while

c) do …while

d) continue

ANS: (a)

4.15 Which of the following is *not* specified by the following code segment:

for (c = 1; c <= 10; c++)

a) initial value of the loop counter

b) loop continuation test

c) increment of the loop counter

d) body statement of the loop

ANS: (d)

4.16 A programmer writes a for statement to count from 1 to 10 and explicitly mentions the 1 and the 10 in the for “header.” Which relational operator would probably be used in the loop-continuation test?

a) >

b) >=

c) <

d) <=

ANS: (d)

4.17 Using an incorrect relational operator or using an incorrect final value of a loop counter in the condition of a while or for statement is a frequent cause of \_\_\_\_\_\_\_\_\_\_errors.

a) syntax

b) compilation

c) off-by-one

d) divide-by-zero

ANS: (c)

4.18 The comma operator is most often used in \_\_\_\_\_\_\_\_\_\_ statements.

a) while

b) do … while

c) for

d) switch

ANS: (c)

4.19 Which statement is *false*?

a) Comma operators evaluate lists of expressions from right to left.

b) The value of a comma-separated list of expressions is the value of the rightmost expression in the list.

c) The type of a comma-separated list of expressions is the type of the rightmost expression in the list.

d) The comma operator is often used to specify multiple initializations in one particular type of iteration statement.

ANS: (a)

4.20 What happens if the loop-continuation test is omitted in a for-statement?

a) C assumes the condition is false, so the loop terminates.

b) A syntax error occurs.

c) C assumes the condition is true, so the loop executes one more time, then terminates.

d) An infinite loop.

ANS: (d)

4.21 A programmer intentionally creates a for-loop with the following for header:

for (; ;)

The programmer’s intent was most likely to create

a) a syntax error

b) an infinite loop

c) a logic error

d) a divide-by-zero error

ANS: (b)

4.22 Which of the following is an incorrect expression to increment c by 1 in the increment expression of a for “header?”

a) c += 1

b) ++c

c) c++

d) c + 1 = c

ANS: (d)

**4.5 for Statement: Notes and Observations**

4.23 If the loop-continuation condition in a for statement is initially *false*, \_\_\_\_\_\_\_\_.

(a) the body portion of the loop is not performed

(b) execution proceeds with the statement following the for statement

(c) both a and b

(d) none of the above

ANS: (c)

4.24 If the increment of the for statement is \_\_\_\_\_\_\_\_ then the loop counts \_\_\_\_\_\_\_\_.

(a) true, downwards

(b) false, downwards

(c) positive, downwards

(d) negative, downwards

ANS: (d)

4.25 Which statement regarding for statements is *false*?

a) In a for statement, the initialization, loop-continuation condition, and increment can contain arithmetic expressions.

b) The increment must be greater than zero.

c) If the loop-continuation condition is initially false, the body of the loop is not performed.

d) It is common to use the control variable for controlling iteration while never mentioning it in the body of the loop.

ANS: (b)

4.26 Which statement about a correct for statement with an initialization expression, a loop-continuation test, an increment expression and a loop body is *false*?

a) The initialization expression is executed only once.

b) The loop-continuation test is evaluated each time through the loop.

c) The initialization is performed each time through the loop.

d) The increment expression is performed after the loop body.

ANS: (c)

**4.6 Examples Using the for Statement**

4.27 The for statement header

for (i = 1; i < 100; ++i)

performs the body of the loop for

a) values of the control variable from 1 to 100 in increments of 1.

b) values of the control variable from 1 to 99 in increments of 1.

c) values of the control variable from 0 to 100 in increments of 1.

d) values of the control variable from 0 to 99 in increments of 1.

ANS: (b)

4.28 What is the highest value assumed by the loop counter in a correct for statement with the following header?

for (i = 7; i <= 72; i += 7)

a) 7

b) 77

c) 70

d) 72

ANS: (b)

4.29 What is produced by a for statement with a correct body and with the following header

for (i = 20; i >= 2; i += 2)

a) a syntax error

b) a divide-by-zero error

c) an infinite loop

d) the even values of i from 20 down to 2.

ANS: (c)

4.30 Which statement is generally *false*?

a) Statements preceding a for and statements in the body of a for should typically be merged into the for header.

b) Limit the size of control statement headers to a single line, if possible.

c) Initialization of a for loop control variable can occur before the for loop executes and not in the loop itself.

d) The increment portion of a for header can be a decrement.

ANS: (a)

4.31 Which expression raises x to the y power?

a) x \*\* y

b) x ^ y

c) x pow y

d) pow(x, y)

ANS: (d)

4.32 Which statement is *true*?

a) Use float variables to perform monetary calculations in C.

b) Use double variables to perform monetary calculations C.

c) Monetary calculations can be performed in C.

d) Printing with %.2f guarantees correct monetary calculations in C.

ANS: (c)

4.33 Which statement is *true*?

a) The conversion specifier %7.2f prints a floating-point value with a field width of 10 positions.

b) The conversion specifier %7.2f prints a floating-point value with 7 positions to the left of the decimal point.

c) The conversion specifier %7.2f prints a floating-point value with 5 positions to the left of the decimal point.

d) The conversion specifier %7.2f prints a floating-point value with 4 positions to the left of the decimal point.

ANS: (d)

**4.7 switch Multiple-Selection Statement**

4.34 A switch statement should be used

(a) as a single-selection statement

(b) as a double-selection statement

(c) when a variable may assume many different values which must be tested

(d) to replace all if and if…else statements

ANS: (c)

4.35 In a switch statement

(a) a breakis required after each case

(b) multiple actions do not need to be enclosed in braces

(c) a defaultcaseis required

(d) a breakis required after the defaultcase

ANS: (b)

4.36 Which statement regarding the switch statement is *false*?

a) It’s appropriate for algorithms that contain a series of decisions in which a variable or expression is tested separately for each of the constant integral values it may assume.

b) The default case is required.

c) The default case must be at the bottom of the switch after all the non-default cases.

d) Many cases may all invoke the same code.

ANS: (b)

4.37 Which statement is *true*?

a) EOF must have the value 1 on all C systems.

b) EOF is a symbolic constant defined in the <symbol.h> header file.

c) EOF is a symbolic variable defined in the <stdio.h> header file.

d) EOF is a symbolic integer constant defined in the <stdio.h> header file.

ANS: (d)

**4.8 do…while Iteration Statement**

4.38 If a do…while statement is used,

(a) an infinite loop will not take place

(b) the counter must be preincremented if it’s also the condition

(c) the body of the loop will execute at least once

(d) an off-by-one error will not occur

ANS: (c)

4.39 The program segment

int counter = 1;

do {

printf("%i ", counter);

} while (++counter <= 10);

will \_\_\_\_\_\_\_\_.

(a) print the numbers 1 through 11

(b) print the numbers 1 through 10

(c) print the numbers 1 through 9

(d) cause a syntax error

ANS: (b)

4.40 Which statement is *true*?

a) The do … while iteration statement is an alternate notation for the while iteration statement; these statements function identically.

b) The do … while iteration statement tests the loop-continuation condition *before* the loop body is performed.

c) The loop body of a correct do … while iteration statement is always executed at least once.

d) The braces delineating the body of a do … while statement are always required.

ANS: (c)

**4.9 break and continue Statements**

4.41 Which of the following is *false*?

(a) break and continue statements alter the flow of control.

(b) continue statements skip the remaining statements in the body of the loop in which they are embedded.

(c) break statements exit from the loop in which they are embedded.

(d) a continue statement can never appear in the else part of an if statement.

ANS: (d)

4.42 Which statement is *true*?

a) The break statement causes an immediate exit from a while, for, do … while or if … else statement.

b) The continue statement is designed for use with the while, for, do … while or switch statements.

c) An equivalent while statement for any for statement can always be formed.

d) The break statement causes an immediate exit from a while, for, do … while or switch statement.

ANS: (d)

**4.10 Logical Operators**

4.43 In C, the condition 4 > y > 1

(a) evaluates correctly

(b) does not evaluate correctly and should be replaced by (4 > y && y > 1)

(c) does not evaluate correctly and should be replaced by (4 > y & y > 1)

(d) does not evaluate correctly and should be replaced by (4 > y || y > 1)

ANS: (b)

4.44 The OR (||) operator

(a) has higher precedence than the AND (&&) operator

(b) stops evaluation upon finding one condition to be *true*

(c) associates from right to left

(d) is a ternary operator

ANS: (b)

4.45 The condition

num != 65

*cannot* be replaced by:

(a) num > 65 || num < 65

(b) !(num == 65)

(c) num – 65

(d) !(num – 65)

ANS: (d)

4.46 An operator that associates from right to left is

(a) !=

(b) ,

(c) ()

(d) ?:

ANS: (d)

4.47 An example of a unary operator is

(a) a relational operator

(b) an assignment operator

(c) an increment operator

(d) a logical operator

ANS: (c)

4.48 Which statement is *true*?

a) To test multiple conditions in the process of making a decision requires logical operators.

b) The keywords for the logical operators are AND, OR and NOT.

c) The logical AND of two expressions is true if and only if *each* of the conditions is true.

d) Truth tables deal only with cases in which all conditions are truthful (i.e., true).

ANS: (c)

4.49 Which statement is *true*?

a) Operator || has a higher precedence than operator &&.

b) In expressions involving operator ||, making the condition that is most likely to be false the leftmost condition can often reduce execution time.

c) The logical negation operator is a binary operator.

d) In expressions using operator &&**,** making the condition that is most likely to be false the leftmost condition can often reduce execution time.

ANS: (d)

**4.11 Confusing Equality (==) and Assignment (=) Operators**

4.50 Non-constant variables are also known as

(a) *lvalues*, but can be used as *rvalues*

(b) *lvalues*, and can not be used as *rvalues*

(c) *rvalues*, but can be used as *lvalues*

(d) constant variables

ANS: (a)

4.51 Consider the following code, assuming that x is an integer variable with an initial value of 12:

if (x = 6) {

printf("%i", x);  
}

What is the output?

(a) 6

(b) 12

(c) nothing

(d) a syntax error is produced

ANS: (a)

4.52 Of the following, which is *not* a logic error?

(a) Using the assignment (=) operator instead of the (==) equality operator to determine if two values are equal

(b) Dividing by zero

(c) Failing to initialize counter and total variables before the body of a loop

(d) Using commas instead of the two required semicolons in a for header

ANS: (d)

4.53 Which statement is *false*?

a) Any expression in C that produces a value can be used in the decision portion of any control statement.

b) When tested for truth or falsity, an expression that produces a nonzero value is treated as true.

c) Assignments in C produce a value, namely the value that the left-hand side of the assignment had prior to the assignment.

d) Operator == is for comparisons; operator = is for assignment.

ANS: (c)

**4.12 Structured Programming Summary**

4.54 The \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_, and \_\_\_\_\_\_\_\_\_\_\_\_ are the only three forms of control necessary.

(a) switch, if, else

(b) sequence, selection, iteration

(c) break, continue, if…else

(d) for, while, do…while

ANS: (b)

4.55 Which statement is *true*?

a) Connecting flowchart symbols arbitrarily always forms structured programs.

b) In a structured program, control statements can only be stacked or sequenced.

c) In the “Rules for Forming Structured Programs (and Structured Flowcharts),” the rule that states, “Any rectangle (action) can be replaced by any control statement” is called the “nesting rule.”

d) Structured programming does *not* improve the program-development process.

ANS: (c)

**4.13 Secure C Programming**

4.56 Which of the following statements is true?

(a) Function scanf does not return a value.

(b) You should never check the return value of function scanf.

(c) You should check the return value of function scanf to ensure that the value it returns matches the number of items that should have been input.

(d)None of the above.

ANS: (c)

4.57 Which of the following statements is true?

(a) If the return value of function scanf matches the number of items that should have been input, then all the inputs are valid.

(b) Even if a scanf operates successfully, the values read might still be invalid.

(c) When a program expects to receive input values in a specific range, you should peform range checking on the inputs to ensure that the values received are indeed in that range (e.g., in a program that expects grades in the range 0-100, you should check that every grade is in that range).

(d)Both (b) and (c).

ANS: (d)