**6.1 Introduction**

6.1 Arrays are data structures consisting of related data items of the same \_\_\_\_\_\_\_\_\_\_.

a) sort order

b) subscript

c) type

d) element

ANS: (c)

6.2 Arrays and structures are \_\_\_\_\_\_\_\_\_\_ entities in that they remain the same size throughout program execution.

a) dynamic

b) automatic

c) register

d) static

ANS: (d)

6.3 Lists, queues, stacks and trees are \_\_\_\_\_\_\_\_\_\_ data structures that may grow and shrink as programs execute.

a) flexible

b) automatic

c) dynamic

d) static

ANS: (c)

**6.2 Arrays**

6.4 An array is *not* \_\_\_\_\_\_\_\_.

(a) a consecutive group of memory locations

(b) indexed by integers

(c) pointer-based

(d) a dynamic entity

ANS: (d)

6.5 Which of the following is *false*?

(a) the first element of an array is the zeroth

(b) the last element of an array is the array size - 1

(c) the position number contained within square brackets is called a subscript

(d) a subscript cannot be an expression.

ANS: (d)

6.6 An array is a group of memory locations related by the fact that they all have \_\_\_\_\_\_\_\_\_\_ array name and \_\_\_\_\_\_\_\_\_\_ type.

a) different, different

b) same, different

c) different same

d) same, same

ANS: (d)

6.7 The first element in every array is the \_\_\_\_\_\_\_\_\_\_ element.

a) null

b) 1

c) 0

d) empty

ANS: (c)

6.8 Which statement is *false*?

a) The brackets used to enclose the subscript of an array are not an operator in C.

b) To refer to a particular element in an array, we specify the name of the array and the position number of the element in the array.

c) The position number within an array is more formally called a subscript.

d) “Array element seven” and the “seventh element of an array” do not mean the same thing. This is a frequent source of off-by-one errors.

ANS: (a)

**6.3 Defining Arrays**

6.9 Which statement would be used to define a 10 element integer array c?

(a) Array c = int[10];

(b) c = int[10];

(c) int Array c[10];

(d) int c[10];

ANS: (d)

6.10 Which definition tells the computer to reserve 12 elements for integer array c?

a) c[12] int;

b) int c [11];

c) c[11] int;

d) int c[12];

ANS: (d)

**6.4 Array Examples**

6.11 Which of the following is *not* a correct way to initialize an array?

(a) int n[5] = {0, 7, 0, 3, 8, 2};

(b) int n[] = {0, 7, 0, 3, 8, 2};

(c) int n[5] = {7};

(d) int n[5] = {6, 6, 6};

ANS: (a)

6.12 Constant variables

(a) can be assigned values in executable statements

(b) do not have to be initialized when they are defined

(c) can be used to specify array sizes, thereby making programs more scalable

(d) can be used to specify array sizes, but this makes programs harder to understand

ANS: (c)

6.13 Referencing elements outside the array bounds

(a) can result in changes to the value of an unrelated variable

(b) is impossible because C checks to make sure it does not happen

(c) is a syntax error

(d) enlarges the size of the array

ANS: (a)

6.14 Strings can not

(a) be initialized using string literals

(b) end in a character other than the null character

(c) be initialized with initializer lists

(d) be treated as arrays of characters

ANS: (b)

6.15 Assume string1 is a character array. Which of the following operations does *not* produce a string?

(a) string1[] = "test";

(b) string1[] = {'t', 'e', 's', 't', '\0'};

(c) string1[] = {'t', 'e', 's', 't'};

(d) string1[] = " ";

ANS: (c)

6.16 Suppose a program contains the code

for (i = 1; i < = 10; i++) {

n[i] = 0;   
 }

Which statement about this code must be *true*?

a) It contains an off-by-one error.

b) It contains a syntax error.

c) It is initializing the first 10 elements of an array.

d) It is initializing successive elements of an array.

ANS: (d)

6.17 What’s wrong with this code?

int[] = (1, 2, 3, 4, 5);

a) The array size must be specified in the square brackets.

b) The parentheses should be square brackets.

c) The square brackets should be curly braces.

d) The parentheses should be curly braces.

ANS: (d)

6.18 If there are fewer initializers than elements in the array, the remaining elements are \_\_\_\_\_\_\_\_\_\_.

a) deleted

b) ignored

c) initialized to empty

d) initialized to zero

ANS: (d)

6.19 Which statement is *true*?

a) Arrays are automatically initialized to zero.

b) To initialize all array elements to zeros, the array definition must explicitly initialize each element to zero.

c) Array elements can be initialized to zero by the array definition only at compile time.

d) Definitions for automatic arrays initialize the arrays at execution time.

ANS: (d)

6.20 The following array definition

int n[5] = {32, 27, 64, 18, 95, 14};

a) is correct

b) causes a syntax error because there are only five initializers and six array elements.

c) causes a logic error because there are only five elements but there are six initializers.

d) causes a syntax error because there are six initializers but only five array elements.

ANS: (d)

6.21 Which statement is *true*?

a) A symbolic constant is an identifier that is replaced with replacement text by the C preprocessor after the program is compiled.

b) Using symbolic constants to specify array sizes makes programs run faster.

c) Preprocessor directives are not C statements.

d) The #define preprocessor directive must end in a semicolon.

ANS: (c)

6.22 Which statement is *true*?

a) Assigning a value to a symbolic constant in an executable statement is a syntax error.

b) A symbolic constant is a variable.

c) Space is reserved for both symbolic constants and variables that hold values at execution time.

d) Only uppercase letters can be used for symbolic constant names.

ANS: (a)

6.23 Which statement is *true* regarding the statement

++frequency[responses[answer]];

a) This statement increases the appropriate frequency counter depending on the value of responses[answer].

b) This statement increases the appropriate answer counter depending on the value of frequency[responses].

c) This statement increases the appropriate responses counter depending on the value of frequency[answer].

d) This statement produces a syntax error because subscripts cannot be nested.

ANS: (a)

6.24 Which statement is *false*?

a) C has no built-in bounds checking to prevent the computer from referring to an array element that does not exist.

b) The programmer is responsible for implementing array bounds checking.

c) Referring to an element outside the array bounds is a syntax error.

d) Referring to an element outside the array bounds is a logic error.

ANS: (d)

**6.5 Using Character Arrays to Store and Manipulate Strings**

6.25 Which statement is *false*?

a) The string termination character is called the null character.

b) The string “string” actually occupies 7 characters in memory.

c) The character representation of the last character in memory of a string is ‘\0’.

d) A character array representing a string should be defined as large as the actual number of characters in the string minus the terminating character; C automatically provides the space for the terminating character.

ANS: (d)

6.26 The definition

char string1[] = "first";

is equivalent to:

a) character string1[] = {'f', 'i', 'r', 's', 't', '\0'};

b) char string1 = {'f', 'i', 'r', 's', 't', '\0'};

c) char string1[] = {'f', 'i', 'r', 's', 't'};

d) char string1[] = {'f', 'i', 'r', 's', 't', '\0'};

ANS: (d)

6.27 Which statement is *false*?

a) Function scanf reads characters into memory from the keyboard until the first input whitespace character is encountered.

b) Function scanf can write beyond the boundary of the array into which input is being placed.

c) Function printf does not care how large the array it is printing is.

d) When using scanf, you must always precede with the & operator the name of each variable into which inputs are being placed.

ANS: (d)

**6.6 Static Local Arrays and Automatic Local Arrays**

6.28 Which statement is *false*?

a) A static local variable exists for the duration of the program.

b) A static local variable is visible only in the control structure in which it is defined.

c) A static local array is not created and destroyed each time the function is entered and exited, respectively.

d) Arrays that are defined static are automatically initialized once at compile time.

ANS: (b)

**6.7 Passing Arrays to Functions**

6.29 Unless otherwise specified, entire arrays are passed \_\_\_\_\_\_\_\_\_\_ and individual array elements are passed \_\_\_\_\_\_\_\_\_\_.

(a) call-by-value, call-by-reference

(b) call-by-reference, call-by-value

(c) call-by-value, call-by-value

(d) call-by-reference, call-by-reference

ANS: (b)

6.30 Which of the following is false about a function being passed an array?

(a) it knows the size of the array it was passed

(b) it is passed the address of the first element in the array

(c) it is able to modify the values stored in the array

(d) all of the above are true

ANS: (a)

6.31 To prevent modification of array values in a function,

(a) the array must be defined staticin the function.

(b) the array parameter can be preceded by the constqualifier.

(c) a copy of the array must be made inside the function.

(d) the array must be passed call-by-reference.

ANS: (b)

6.32 To pass an array to a function, specify

a) the name of the array without any brackets.

b) the name of the array preceded by an empty pair of brackets.

c) the name of the array followed by a pair of brackets including the size of the array.

d) the name of the array followed by a pair of brackets including a number one less than the size of the array.

ANS: (a)

6.33 Which statement is *false*?

a) C automatically passes arrays to functions using simulated call by reference.

b) When C passes an array to a function, the called function normally can modify the element values in the caller’s original array.

c) The name of an array is actually the address of the first element of the array.

d) When an array is passed in a function call, the calling function knows precisely where in the called function’s memory the passed array is.

ANS: (d)

6.34 The special conversion specifier for printing addresses is \_\_\_\_\_\_\_\_\_\_.

a) %a

b) %m

c) %p

d) %loc

ANS: (c)

6.35 Which statement is *true*?

a) Entire arrays are passed simulated call by reference and individual array elements are normally passed simulated call by reference.

b) Entire arrays are passed simulated call by reference and individual array elements are normally passed call by value.

c) Entire arrays are passed call by value and individual array elements are normally passed simulated call by reference.

d) Entire arrays are passed call by value and individual array elements are normally passed call by value.

ANS: (b)

**6.8 Sorting Arrays**

6.36 Which statement about bubble sort is *true*?

(a) a maximum of *n* passes are needed to sort the array, where *n* is the number of elements

(b) swapping values requires two assignments

(c) performance is maximized

(d) the algorithm is very simple compared to other sorting procedures

ANS: (d)

6.37 Which statement about the bubble sort is *false*?

a) It is easy to program.

b) It is a high-performance sort.

c) It compares only adjacent elements with one another.

d) The bubble sort compares successive pairs of elements.

ANS: (b)

6.38 A bubble sort of 1000 elements requires a maximum of \_\_\_\_\_\_\_\_\_\_ passes.

a) 1001

b) 1000

c) 999

d) 998

ANS: (c)

**6.9 Case Study: Computing Mean, Median and Mode Using Arrays**

6.39 In order to calculate the \_\_\_\_\_\_\_\_\_\_ of an array of values, the array must be sorted.

(a) median

(b) mode

(c) mean

(d) (a), (b), and (c)

ANS: (a)

6.40 The \_\_\_\_\_ is the average value of a set of data items.

a) mean

b) median

c) mode

d) matrix

ANS: (a)

6.41 Calculating which of the following normally requires the data to be sorted first

a) mean

b) median

c) mode

d) total

ANS: (b)

6.42 The \_\_\_\_\_\_\_\_\_\_ is the value that occurs most frequently in the data.

a) mean

b) median

c) mode

d) master

ANS: (c)

**6.10 Searching Arrays**

6.43 The binary search technique

(a) is better suited to small arrays

(b) is better suited to unsorted arrays

(c) can only be used on a sorted array

(d) is slower than a linear search

ANS: (c)

6.44 The maximum number of comparisons needed for the binary search of a 2000 element array is

(a) 9

(b) 15

(c) 11

(d) 14

ANS: (c)

6.45 Which of these is generally thought of as a high-performance technique?

a) bubble sort

b) linear search

c) binary search

d) iteration

ANS: (c)

6.46 Which of the following statements is *false*?

a) The linear searching method works well for small arrays.

b) The linear searching method works well for unsorted arrays.

c) The binary search algorithm eliminates from consideration one half of the elements in a sorted array after each comparison.

d) The binary search terminates only when the search key is equal to the middle element of a subarray.

ANS: (d)

6.47 A sorted array of a million elements can be searched by a binary search in \_\_\_\_\_\_\_\_\_\_ or fewer comparisons.

a) 10

b) 20

c) 30

d) 999,999

ANS: (b)

**6.11 Multidimensional Arrays**

6.48 A two-dimensional array element incorrectly referenced as a[x, y] is actually evaluated as

(a) a[x][y]

(b) a[y]

(c) a[x]

(d) a[0]

ANS: (b)

6.49 Given the following definitions, what is the value of b[1][0]?

int b[2][2] = {{1}, {3, 4}};

(a) 0

(b) 1

(c) 3

(d) this isn’t a valid definition

ANS: (c)

6.50 Which of the following does *not* initialize all of the array elements to 0?

(a)   
int b[2][2];

b[0][0] = b[0][1] = b[1][0] = b[1][1] = 0;

(b) int b[2][2] = {0};

(c)   
int b[2][2];

for (int i = 0; i < 2; ++i) {

for (int j = 0; j < 2; ++j) {

b[i][j] = 0;  
 }  
}

(d) all of the above initialize all of their elements to 0.

ANS: (d)

6.51 Which statement is *false* about multidimensional arrays?

a) C supports multidimensional arrays.

b) A common use of multidimensional arrays is to arrange data into tables consisting of rows and columns.

c) To identify a particular table element requires two indices.

d) Every ANSI C system can support arrays with as many indices as the programmer chooses.

ANS: (d)

6.52 An array containing 3 columns and 4 rows is typically referred to as a \_\_\_\_\_\_\_\_\_\_.

a) 12-element array

b) 3-by-4 array

c) 13-element array, because of the zero element

d) 4-by-3 array

ANS: (d)

6.53 Which initialization is *not* performed by the following definition?

int b[2][2] = {{1}, {3, 4}};

a)b[0][0] is set to 1

b) b[0][1] is set to 1

c) b[1][0] is set to 3

d) b[1][1] is set to 4

ANS: (b)

**6.12 Variable-Length Arrays**

No questions.

**6.13 Secure C Programming**

6.54 Which of the following statements is *false*?

(a) C provides automatic bounds checking for arrays.

(b) C provides no automatic bounds checking for arrays, so you must provide your own.

(c) Allowing programs to read from or write to array elements outside the bounds of arrays are common security flaws.

(d)Writing to an out-of-bounds element (known as a buffer overflow) can corrupt a program’s data in memory, crash a program and allow attackers to exploit the system and execute their own code.

ANS: (a)

6.55 Assuming myString is a 20-element char array, which of the following statements might result in buffer overflow?

(a) scanf("%19s", myString);

(b) scanf\_s("%s", myString, 20);

(c) scanf\_s("%19s", myString, 20);

(d)scanf("%s", myString);

ANS: (d)