**10.1 Introduction**

10.1 \_\_\_\_\_\_\_\_ may contain different data types.

(a) structures

(b) arrays

(c) both a and b

(d) none of these

ANS: (a)

10.2. \_\_\_\_\_\_\_\_\_\_ are collections of related variables—sometimes referred to as aggregates—under one name.

a) Members

b) Enumerations

c) Structures

d) Files

ANS: (c)

10.3. Structures may contain variables of many different data types—in contrast to \_\_\_\_\_\_\_\_\_\_\_ that contain only elements of the same data type.

a) files

b) arrays

c) constants

d) functions

ANS: (b)

**10.2 Structure Definitions**

10.4 Two structure variables with the *same* member values

(a) always compare equally.

(b) never compare equally.

(c) sometimes compare equally.

(d) only compare equally if both structure variables contain holes.

ANS: (c)

10.5 What does the deck[52] array contain in the following statement?

struct card a, deck[52], \*cPtr;

(a) card structure elements

(b) a structure elements

(c) \*cPtr elements

(d) none of these

ANS: (a)

10.6. Keyword \_\_\_\_\_\_\_\_\_\_ introduces the structure definition.

a) structure

b) str

c) strdef

d) struct

ANS: (d)

10.7. A structure containing a member that’s a pointer to the same structure type is referred to as a \_\_\_\_\_\_\_\_\_\_ structure.

a) self-referential

b) self-describing

c) self-recursive

d) self- elemental

ANS: (a)

10.8. Which statement is *false*?

a) Structures are derived data types.

b) Each structure definition must end with a semicolon.

c) A structure can contain an instance of itself.

d) Structures may not be compared using operators == and !=.

ANS: (c)

10.9. Which of the following is *not* a valid operation on a structure?

a) Assigning structure variables to structure variables of the same type.

b) Taking the address of a structure variable.

c) Using the sizeof operator to determine the size of a structure variable.

d) Comparing structures of the same type with relational operators.

ANS: (d)

**10.3 Initializing Structures**

10.10 Structure variables may *not* be initialized by

(a) setting user-defined default variable values in the struct definition.

(b) assigning values to individual data members.

(c) array-like member initializer lists.

(d) assignment statements.

ANS: (a)

10.11 What does the following statement do?

struct card a = {"Three", "Hearts"};

(a) It creates a variable card of type struct with two members specified in the list.

(b) It creates two variables named Three and Hearts of type struct card a.

(c) It creates a variable a to be of type struct card and initializes it to the values in the list.

(d) It creates two variables named Three and Hearts of type struct card.

ANS: (c)

10.12. If there are fewer \_\_\_\_\_\_\_\_\_\_ in the list than members in the structure, the remaining members are automatically initialized to 0 or NULL.

a) quantifiers

b) initializers

c) numerators

d) variables

ANS: (b)

10.13. Structures can be initialized in \_\_\_\_\_\_\_\_\_\_ statements.

a) assignment

b) null

c) scope

d) empty

ANS: (a)

**10.4 Accessing Structure Members**

10.14 Arrays are

(a) always passed call-by-reference.

(b) passed call-by-reference unless inside a structure.

(c) always passed call-by-value.

(d) passed call-by-value unless inside a structure.

ANS: (b)

10.15 The \_\_\_\_\_\_\_\_ accesses a structure member via the structure variable name.

(a) structure member operator

(b) structure pointer operator

(c) structure arrow operator

(d) none of these

ANS: (a)

10.16. The expression aptr->suit is equivalent to \_\_\_\_\_\_\_\_\_\_.

a) aptr.suit

b) \*aptr.suit

c) (\*aptr).suit

d) \*aptr.(suit)

ANS: (c)

10.17. Which of the following is *false*?

a) Structure member names throughout a program must be unique.

b) Attempting to refer to a member of a structure by using only the member’s name is an error.

c) Inserting a space between the - and > of operator -> is an error.

d) The dot operator (.) is the structure member operator.

ANS: (a)

**10.5 Using Structures with Functions**

10.18 Structures may be passed to functions by \_\_\_\_\_\_\_\_.

(a) passing individual structure members

(b) passing an entire structure

(c) passing a pointer to the structure

(d) all of these

ANS: (d)

10.19 Arrays of structures \_\_\_\_\_\_\_\_.

(a) are automatically passed by reference

(b) are automatically passed by value

(c) cannot be passed by reference

(d) None of the above.

ANS: (a)

10.20. A structure of arrays is automatically passed \_\_\_\_\_\_\_\_\_\_.

a) by value

b) by reference

c) by aggregate

d) by masking

ANS: (a)

10.21. An array of structures is automatically passed \_\_\_\_\_\_\_\_\_\_.

a) by value

b) by reference

c) by aggregate

d) by enumeration

ANS: (b)

10.22. Which of the following is *false*?

a) To pass a structure by reference, pass the address of the structure variable.

b) A way to pass an array by value, is to create a structure with the array as a member then pass the name of the structure.

c) To pass a structure by reference, pass the name of the structure variable.

d) Passing large structures by reference is more efficient than passing large structures by value.

ANS: (c)

**10.6 typedef**

10.23 typedef is used to

(a) create a name that is an alias for a previously defined data type.

(b) create new data types.

(c) cast one struct to another type.

(d) initialize struct members.

ANS: (a)

10.24 Advantages of typedef do *not* include

(a) making programs more portable by allowing data types to be easily changed to meet system specifications.

(b) making type names shorter.

(c) making programs more readable.

(d) increasing the efficiency of accessing struct member variables.

ANS: (d)

10.25 Creating a new name with typedef \_\_\_\_\_\_\_\_\_\_.

a) creates a new type

b) creates a new type name

c) creates a new variable name

d) creates a new variable

ANS: (b)

10.26 Which statement is *true*?

a) Often, typedef is used to create synonyms for the basic data types.

b) Keyword typedef is used to create antonyms (or opposites) for the basic data types.

c) Names for structure types are typically defined with typedef to create longer type names.

d) The first letter of each typedef name must be capitalized.

ANS: (a)

**10.7 Example: High-Performance Card-Shuffling and Dealing Simulation**

10.27 What does the statement typedef struct card Card; do?

a) Defines card as a synonym for Card.

b) Defines Card as a synonym for card.

c) Defines Card as a synonym for struct card.

d) Defines Card as a synonym for typedef struct card.

ANS: (c)

10.28 Which of the following is *always* a syntax error?

a) Not using srand in every program that uses rand.

b) Not using rand in every program that uses srand.

c) Not using const when passing an array.

d) Forgetting to include the array subscript when referring to individual structures in an array of structures.

ANS: (d)

**10.8 Unions**

10.29 Which of the following is *true*?

(a) Unions may be compared using the == operator.

(b) The address operator (&) cannot be used to take the address of a union.

(c) Unions may only contain two data types.

(d) Only one union member, and thus one data type, can be referenced at a time

ANS: (d)

10.30 Which of the following is *not* an operation that can be performed on a union?

(a) comparing using the != operator

(b) taking the address (&) of a union

(c) accessing union members using the structure member operator

(d) assigning a union to another union of the same type

ANS: (a)

10.31 Which of the following is *true*?

a) A union typically makes less efficient use of memory than a struct.

b) A union is another name for a struct.

c) A union is a derived data type whose members share the same storage space.

d) Unions are always portable between different computers with different compilers.

ANS: (c)

10.32 Which statement is *true*?

a) The members of a union can be of any type.

b) The members of a union must all be of the same type.

c) A union may not be assigned to another union of the same type.

d) Unions may be compared to other unions of the same type.

ANS: (a)

**10.9 Bitwise Operators**

10.33 The most basic unit of data on a computer is the

(a) bit.

(b) byte.

(c) file.

(d) int.

ANS: (a)

10.34 Which of the following is *not* a bitwise operator?

(a) ^

(b) &

(c) ~

(d) \*

ANS: (d)

10.35 Let Bit1 = Bit2 = 1. Which of the following does *not* have the same result as the others?

(a) Bit1 & Bit2

(b) Bit1 | Bit2

(c) Bit1^Bit2

(d) ~(~Bit2)

ANS: (c)

10.36 Evaluate (00001000 & 11000101) ^ (11110000)

(a) 00111101

(b) 11000000

(c) 00111101

(d) 11110000

ANS: (d)

10.37 Let x be an int on a machine with four-byte ints. What effect does

x<<=1;

x>>=1;

have?

(a) There is no effect.

(b) The leftmost bit of x is set to zero.

(c) The rightmost bit of x is set to zero.

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

ANS: (b)

10.38 For any eight-bit x, which of the following does *not* result in zero?

(a) x &= (~x)

(b) x ^= x

(c) x <<= 8

(d) x |= x

ANS: (d)

10.39 The bitwise operators can be used to manipulate the bits of variables of type \_\_\_\_\_\_\_\_\_\_.

a) float

b) double

c) long

d) long double

ANS: (c)

10.40 Which operator sets the bits in the result to 1 only when the corresponding bits in the two operands are *both* 1?

a) bitwise AND

b) bitwise inclusive OR

c) bitwise exclusive OR

d) bitwise complement

ANS: (a)

10.41 Which operator sets the bits in the result to 1 if *at least* *one* of the corresponding bits in the two operands is 1?

a) bitwise AND

b) bitwise inclusive OR

c) bitwise exclusive OR

d) bitwise complement

ANS: (b)

10.42 Which operator sets the bits in the result to 1 if at *exactly one* of the corresponding bits in the two operands is 1?

a) bitwise AND

b) bitwise inclusive OR

c) bitwise exclusive OR

d) bitwise complement

ANS: (c)

10.43 Which statement is *true* of the << operator?

a) It’s called the right shift operator.

b) It’s called the bitwise right shift operator.

c) It shifts the bits of the second operand left by the number of bits specified by the first operand.

d) It fills from the right with 0 bits.

ANS: (d)

10.44 Which statement is *true* of the >> operator?

a) It is called the left shift operator.

b) It is called the bitwise left shift operator.

c) It shifts the bits of the first operand right by the number of bits specified by the second operand.

d) It fills from the left portably across platforms.

ANS: (c)

10.45 In bitwise manipulations, a mask is typically \_\_\_\_\_\_\_\_\_\_.

a) A floating point value with specific bits set to 1.

b) An integer value with specific bits set to 1.

c) An integer value with specific bits set to 0.

d) A floating point value with specific bits set to 0.

ANS: (b)

10.46 Which of the following is *true*?

a) Any bit “ANDed” with 0 yields 0.

b) Any bit “ANDed” with 1 yields 1.

c) Any bit “ANDed” with 0 yields 1.

d) Any bit “ANDed” with 1 yields 0.

ANS: (a)

10.47 Which statement is *true*?

a) || is the bitwise or operator,

b) | is the logical or operator

c) Operators || and | are interchangeable.

d) || and | are each binary operators.

ANS: (d)

10.48 Which statement is *false*?

a) The result of shifting a value is undefined if the right operand is negative.

b) The result of shifting a value is undefined if the right operand has more bits than the number of bits in which the left operand is stored.

c) Left shifting is machine dependent.

d) Right shifting is machine dependent.

ANS: (c)

**10.10 Bit Fields**

10.49 A bitfield must be defined as a \_\_\_\_\_\_\_\_\_\_.

(a) int or unsigned

(b) char

(c) float

(d) long

ANS: (a)

10.50 The number of bits in a bitfield is specified with \_\_\_\_\_\_\_\_\_\_.

(a) parenthesis as in bitfield(4)

(b) a colon as in bitfield : 4

(c) brackets as in bitfield[4]

(d) a dot as in bitfield.4

ANS: (b)

10.51 \_\_\_\_\_\_\_\_\_\_ is not allowed.

(a) Accessing individual bits in a multi-bit bitfield.

(b) Padding a bitfield with bits that cannot be accessed

(c) Having an unnamed bitfield.

(d) Having a bitfield with a zero width.

ANS: (a)

10.52 Which statement is *false*?

a) Bit field manipulations are machine dependent.

b) Bit fields may never cross a word boundary.

c) Bit fields are not “arrays of bits.”

d) It is not possible to take the address of a bit field.

ANS: (b)

10.53 An unnamed bit field with a zero width is used to \_\_\_\_\_\_\_\_\_\_.

a) purge a bit from a bit field

b) initialize a bit field to 0.

c) concatenate two bit fields.

d) align the next bit field on a new storage unit boundary.

ANS: (d)

10.54 Which statement is *false*?

a) Using bit fields can be an effective space saving technique.

b) Using bit fields always results in faster executing machine language.

c) The decision to use bit fields is one of many examples of the kinds of space-time tradeoffs that occur in computer science.

d) An unnamed bit field with a non-zero width is used as padding in a struct of bit fields.

ANS: (b)

**10.11 Enumeration Constants**

10.55 Enumeration constants within an enumeration

(a) must have unique integer values

(b) can be assigned other values once they have been defined

(c) must have unique identifiers

(d) are defined using the keyword const

ANS: (c)

10.56 Which statement is *false*?

a) Assigning a value to an enumeration constant when creating an enumeration type causes a syntax error.

b) An enumeration is a set of integer constants represented by identifiers.

c) Enumeration constant values can be set automatically.

d) Assigning a value to an enumeration constant after it has been defined is a syntax error.

ANS: (a)

10.57 Which statement is *true*?

a) The identifiers in an enumeration must be unique.

b) The constant values assigned to members of an enumeration must be unique.

c) Enumeration constant names must consist of all uppercase letters.

d) Values in an enumeration start with 1 unless specified otherwise.

ANS: (a)

**10.12 Anonymous Structures and Unions**

No questions

**10.13 Secure C Programming**

10.58 Which of the following statements is true?

(a) The size of a struct variable is the sum of its members’ sizes.

(b) struct variables can be compared for equality or inequality.

(c) In a struct variable, undefined extra bytes could contain secure data—left over from prior use of those memory locations—that should not be accessible.

(d)None of the above.

ANS: (c)

10.59 Which of the following statements is true?

(a) Allowing multiple enumeration constants to have the same value can result in difficult-to-find logic errors.

(b) Performing bitwise operations on integer types smaller than int always yields correct results.

(c) Bitwise operations on signed and unsigned integer types are portable across platforms.

(d)None of the above.

ANS: (a)