**12.1 Introduction**

12.1 \_\_\_\_\_\_\_\_\_\_ are collections of data items “lined up in a row”—insertions and deletions are made anywhere in a \_\_\_\_\_\_\_\_\_\_.

a) Linked lists, linked list

b) Queues, queue

c) Stacks, stack

d) Binary trees, binary tree

ANS: (a)

12.2 \_\_\_\_\_\_\_\_\_\_ are important in compilers and operating systems—insertions and deletions are made only at one end of a \_\_\_\_\_\_\_\_\_\_—its top.

a) Linked lists, linked list

b) Queues, queue

c) Stacks, stack

d) Binary trees, binary tree

ANS: (c)

12.3 \_\_\_\_\_\_\_\_\_\_ represent waiting lines; insertions are made at the back (also called the tail) and deletions are made from the front (also called the head) of a \_\_\_\_\_\_\_\_\_\_.

a) Linked lists, linked list

b) Queues, queue

c) Stacks, stack

d) Binary trees, binary tree

ANS: (b)

12.4 \_\_\_\_\_\_\_\_\_\_ facilitate high-speed searching and sorting of data, efficient elimination of duplicate items and compiling expressions into machine language.

a) Linked lists

b) Queues

c) Stacks

d) Binary Trees

ANS: (d)

**12.2 Self-Referential Structures**

12.5 A self-referential structure contains a \_\_\_\_\_\_\_\_ member that points to \_\_\_\_\_\_\_\_.

(a) integer, a structure of the same structure type

(b) pointer, an integer

(c) integer, an integer

(d) pointer, a structure of the same structure type

ANS: (d)

12.6 A(n) \_\_\_\_\_\_\_\_\_\_ pointer normally indicates the end of a data structure.

a) uninitialized

b) NULL

c) self

d) dereferenced

ANS: (b)

**12.3 Dynamic Memory Allocation**

12.7 A \_\_\_\_\_\_\_\_\_\_ occurs when dynamically allocated memory is *not* returned when it’s no longer needed.

(a) memory leak

(b) self-referential error

(c) allocation error

(d) sizeoferror

ANS: (a)

12.8 \_\_\_\_\_\_\_\_\_\_ memory allocation is the ability for a program to obtain more memory space at execution time and to release space no longer needed.

a) Static

b) Active

c) Selective

d) Dynamic

ANS: (d)

12.9 Function malloc takes as an argument the number of bytes to be allocated, and returns a pointer of type \_\_\_\_\_\_\_\_\_\_ to the allocated memory.

a) char \*

b) int \*

c) void \*

d) NULL \*

ANS: (c)

12.10 If no memory is available mallocreturns a(n) \_\_\_\_\_\_\_\_\_\_ pointer.

a) self

b) NULL

c) void

d) empty

ANS: (b)

12.11 Which of the following statements is *true*?

a) A structure’s size is sometimes smaller than the total of the sizes of its members.

b) A structure’s size is always larger than the total of the sizes of its members.

c) A structure’s size is not necessarily the sum of the sizes of its members.

d) A structure’s size is the sum of the sizes of its members.

ANS: (c)

12.12 Which is correct?

a) Use the size operator to determine the size of a structure.

b) Use the struct size operator to determine the size of a structure.

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

d) Determine the size of a structure manually by carefully adding up the sizes of the members.

ANS: (c)

12.13 Not returning dynamically allocated memory when it is no longer needed can cause a system to run out of memory prematurely. This is called a(n) \_\_\_\_\_\_\_\_\_\_.

a) outage

b) memory hole

c) memory access violation

d) memory leak

ANS: (d)

12.14 When memory allocated with malloc is no longer needed, return that memory to the system immediately with \_\_\_\_\_\_\_\_\_\_.

a) free\_memory

b) free\_storage

c) return

d) free

ANS: (d)

12.15 Which of these is *not* a common programming error?

a) Referring to memory that has been freed.

b) Freeing memory (with free) that was not dynamically allocated.

c) Assuming that the size of a structure is simply the sum of the sizes of its members.

d) Calling malloc in a statement without using sizeof.

ANS: (d)

**12.4 Linked Lists**

12.18 \_\_\_\_\_\_\_\_\_\_ is *not* an advantage of linked lists when compared to arrays.

(a) Dynamic memory allocation

(b) Efficient insertion and deletion

(c) Direct access to any list element

(d) Efficient use of memory

ANS: (c)

12.19 For a non-empty linked list, select the code that should appear in a function that adds a node to the end of the list. newPtris a pointer to the new node to be added, and lastPtris a pointer to the current last node. Each node contains a pointer nextPtr, a link to a node.

(a)   
lastPtr->nextPtr = newPtr;

lastPtr = newPtr;

(b)   
lastPtr = newPtr;

lastPtr->nextPtr = newPtr;

(c)   
newPtr->nextPtr = lastPtr;

lastPtr = newPtr;

(d)   
lastPtr = newPtr;

newPtr->nextPtr = lastPtr;

ANS: (a)

12.20 How many pointers are contained in a circular, doubly linked list with five nodes?

(a) 5

(b) 8

(c) 15

(d) 10

ANS: (d)

12.21 A linked list is a \_\_\_\_\_\_\_\_\_\_ collection of self-referential structures, called nodes, connected by pointer links.

a) hierachical

b) linear

c) branching

d) constant

ANS: (b)

12.22 Which of the following is a *non-linear* data structure?

a) linked list

b) queue

c) binary tree

d) stack

ANS: (c)

12.23 Which of the following is *false*?

a) Lists of data can be stored in arrays.

b) The length of a linked list can vary dynamically.

c) Arrays can become full.

d) Linked lists cannot become full.

ANS: (d)

12.24 Which of the following is *false*?

a) Arrays can be maintained in sorted order.

b) Linked lists can be maintained in sorted order.

c) Insertion and deletion in a sorted array (while maintaining sorted order) is efficient.

d) Once the insertion point or the node to be deleted has been located, insertion or deletion in a sorted linked list (while maintaining sorted order) is efficient.

ANS: (c)

12.25 Which statement is *false*?

a) Arrays are normally stored contiguously in memory.

b) Dynamic memory allocation can sometimes use memory more efficiently than using fixed-size data structures.

c) Dynamic memory allocation incurs execution-time overhead.

d) Linked lists are normally stored contiguously in memory.

ANS: (d)

12.26 Functions such as isEmpty and isFull that test a condition and return a value that can be interpreted as true or false, are called \_\_\_\_\_\_\_\_\_\_ functions.

a) imperative

b) declarative

c) predicate

d) conditional

ANS: (c)

12.27 Passing a pointer to a pointer is called \_\_\_\_\_\_\_\_\_\_.

a) double direction

b) pointer passing

c) double indirection

d) indirection

ANS: (c)

**12.5 Stacks**

12.28 Which of the following statements about stacks is *incorrect*?

(a) stacks can be implemented using linked lists.

(b) stacks are first in, first-out (FIFO) data structures.

(c) new nodes can only be added to the top of the stack.

(d) the last node (the bottom) of a stack has a null (zero) link.

ANS: (b)

12.29 A stack is initially empty, then the following commands are performed.

push 5

push 7

pop

push 10

push 5

pop

Which of the following is the correct stack (assume the top of the stack is on the left).

(a) 5 10 7 5

(b) 5 10

(c) 7 5

(d) 10 5

ANS: (d)

12.30 New nodes can be added to a stack and removed from the stack only at its top. For this reason a stack is referred to as a \_\_\_\_\_\_\_\_\_\_ data structure.

a) first-in, first-out

b) linear

c) last-in, first-out

d) dynamic

ANS: (c)

12.31 The link member in the last node of a stack is typically set to \_\_\_\_\_\_\_\_\_\_ indicate the bottom of the stack.

a) void

b) void \*

c) NULL

d) empty

ANS: (c)

12.32 Which of the following statements is *false*?

a) The primary functions used to manipulate a stack are push and pop.

b) Function pop removes a node from the bottom of the stack.

c) Function push creates a new node and places it on top of the stack.

d) A stack can be implemented as a constrained version of a linked list by allowing insertions and deletions only at one end of the linked list.

ANS: (b)

12.33 Which is *not* a popular application of stacks?

a) enabling called functions to return to their callers

b) supporting recursive function calls

c) containing the space created for automatic variables

d) maintaining waiting lines

ANS: (d)

**12.6 Queues**

12.34 A queue receives the following commands (in pseudo-code):

*enqueue 4, 6, 8, 3, 1*

*dequeue three elements*

*enqueue 3, 1, 5, 6*

*dequeue two elements*

What number is at the front of the queue?

(a) 3

(b) 4

(c) 5

(d) 6

ANS: (a)

12.35 A linked list has the functions insertAtFront, removeFromFront, insertAtBack, and removeFromBack, which perform operations on nodes exactly as their names describe. Which two functions would most

naturally model the operation of a queue?

(a) insertAtBackand removeFromBack.

(b) insertAtBackand removeFromFront.

(c) insertAtFrontand removeFromFront.

(d) insertAtFrontand removeFromBack.

ANS: (b)

12.36 Queues are linear data structures with the property that queue nodes are inserted only at the tail of the queue and removed only from the head of the queue. For this reason, queues are referred to as \_\_\_\_\_\_\_\_\_\_ data structures.

a) first-in, first-out

b) first-in, last-out

c) last-in, first-out

d) first-come, first-served

ANS: (a)

12.37 Which of the following is not true of queues?

a) Network packets wait in queues for service at routers .

b) The entry at the front (or head) of the queue is the next to be removed.

c) Queues are used to support print spooling.

d) Queues are used to support high-speed sorting algorithms.

ANS: (d)

**12.7 Trees**

12.38 Select the *incorrect* statement. Binary trees (regardless of the order in which the values are inserted into the tree)

(a) always have multiple links per node.

(b) can be sorted efficiently.

(c) always have the same shape for a particular set of data.

(d) are nonlinear data structures.

ANS: (c)

12.39 Add the following nodes to a binary search tree in the order they appear.

6 34 17 19 16 10 23 3

What is the output of a postorder traversal of this tree?

(a) 3 10 16 23 19 17 34 6

(b) 3 6 17 16 10 19 23

(c) 6 3 34 17 16 10 19 23

(d) 10 16 23 19 17 34 3 6

ANS: (a)

12.40 Suppose you have a list of names sorted in alphabetical order, already stored in one of the data types below. The easiest way to print the names in reverse alphabetical order would be to use a

(a) binary search tree

(b) stack

(c) queue

(d) circular, singly linked list

ANS: (b)

12.41 If you have a 1000-element balanced binary search tree, what is the maximum number of comparisons that may be needed to find an element in the tree?

(a) 500

(b) 10

(c) 20

(d) 8

ANS: (b)

12.42 Which statement about trees is *false*?

a) A tree is a non-linear, two-dimensional data structure.

b) Tree nodes contain two or more links.

c) Binary tree nodes contain two or fewer links.

d) Binary tree nodes contain exactly two links.

ANS: (c)

12.43 Which statement is not *true* for binary trees.

a) The left child in the root node is the first node in the left subtree.

b) The children of a node are called siblings.

c) A node with no children is called an orphan.

d) The root node is the first node in the tree.

ANS: (c)

12.44 Which of the following statements about binary search trees with no duplicate values is *false*?

a) The values in any left subtree are less than the values in its parent node.

b) The values in any right subtree are less than the values in its parent node.

c) The shape of the tree that corresponds to a particular set of data can vary based on the order in which the values are inserted into the tree.

d) It is possible that a binary tree could contain all its values along one straight path through the tree.

ANS: (b)

12.45 A node can only be inserted \_\_\_\_\_\_\_\_\_\_ in a binary search tree.

a) as the root node

b) as a leaf node

c) as a parent node

d) as an ancestor node

ANS: (b)

12.46 The steps for an *in-order traversal* of a binary search tree include each of the following except \_\_\_\_\_\_\_\_\_.

a) Traverse the left subtree in-order.

b) Process the value in the root node.

c) Skip over duplicate values.

d) Traverse the right subtree in-order,

ANS: (c)

12.47 Which type of binary search tree traversal processes the node values in ascending order?

a) in-order traversal

b) pre-order traversal

c) post-order traversal

d) duplicate elimination traversal

ANS: (a)

12.48 Which of the following statements about binary search trees is *false*?

a) The binary search tree facilitates duplicate elimination.

b) In a tightly packed binary search tree, each level contains about half as many elements as the previous level. (The previous level is the level closer to the root node.)

c) When searching a tightly packed billion-element search tree, only about 30 elements (or fewer) are required to locate most elements.

d) When searching a tightly packed million-element search tree, only about 20 elements (or fewer) are required to locate most elements.

ANS: (b)

12.49 Which statement about the level-order traversal of a binary tree is false?

a) It visits the nodes of a tree row by row.

b) The search begins at the root node.

c) The search begins at the row of the leftmost leaf node.

d) On each level of the tree, the nodes are visited left to right.

ANS: (c)

**12.8 Secure C Programming**

12.50 Which of the following statements is *false*?

(a) Pointers should not be left uninitialized.

(b) When you use free to deallocate dynamically allocated memory, the pointer passed to free is set to NULL.

(c) Undefined behavior occurs when you attempt to use free to deallocate dynamic memory that was already deallocated

(d)Function malloc returns NULL if it’s unable to allocate the requested memory.

ANS: (b)