1. A node, in its simplest form, holds how many pieces of data?
   (a) 2
   (b) 4
   (c) 3
   (d) 1

2. Suppose you have three nodes linked together. How many next pointers are null?
   (a) 1
   (b) 2
   (c) 3
   (d) 0

3. Suppose you have three nodes linked together. How many next pointers are not null?
   (a) 3
   (b) 0
   (c) 1
   (d) 2

4. Using the node accessors in the text, how would you swap the values of two nodes, a and b?
   (a) t = a; setNodeNext(a,b); setNodeNext(a,t));
   (b) t = getNodeValue(a);
       setNodeValue(a,getNodeValue(b));
       setNodeValue(b,t);
   (c) t = a; a = b; b = t;
   (d) setNodeValue(b,getNodeValue(a));
       setNodeValue(a,getNodeValue(b));

5. Using the node pointers, how would you swap the values of two nodes, a and b?
   (a) temp = a; a->value = b; b->value = temp;
   (b) temp = a->value;
       a->value = b->value;
       b->value = temp;
   (c) temp = a; a = b; b = a;
   (d) b->value = a->value; a->value = b->value;

6. Suppose you wish to link nodes a and b such that a follows b. Using node accessors from the text, which action accomplishes this wish?
   (a) getNodeNext(b,a);
   (b) getNodeNext(a,b);
   (c) setNodeNext(b,a);
   (d) a = b;

7. Suppose you wish to link nodes a and b such that a follows b. Which action accomplishes this wish?
   (a) b->next = a;
   (b) b = a->next;
   (c) a = b->next;
   (d) a = b;
8. Consider:

\[ z = \text{newNode}("a", \text{newNode}("c", \text{newNode}("b", 0))) \]

Which representation is consistent with \( z \)?

(a) \{a\}{c\}{b\}{0}
(b) \{0\}{b\}{c\}{a}
(c) \{b\}{c\}{a}
(d) \{a\}{c\}{b}

9. Consider:

\[ z = \text{newNode}(2,0); z = \text{newNode}(7,z); z = \text{newNode}(5,z); \]

Which representation is consistent with \( a \)?

(a) \{0\}{2\}{7\}{5\}
(b) \{5\}{7\}{2\}
(c) \{2\}{7\}{5\}
(d) \{5\}{7\}{2\}{0\}

10. Consider:

\[ a = \text{newNode}(2,0); b = \text{newNode}(7,a); c = \text{newNode}(5,b); \]

setNodeNext(a,c);

Which representation is consistent with \( a \)?

(a) \{5\}{7\}{2\}{5\}{0\}
(b) \{2\}{5\}{7\}{2\}
(c) \{5\}{7\}{2\}{2\}{7\}{2\}{5\}{7\}... (goes on forever)
(d) \{5\}{7\}{2\}{5\}{0\}
(e) \{2\}{5\}{7\}{2\}{5\}{7\}{2\}{5}... (goes on forever)

11. Consider:

\[ a = \text{newNode}(2,0); b = \text{newNode}(7,a); c = \text{newNode}(5,b); \]

a->next = c;

Which representation is consistent with \( a \)?

(a) \{5\}{7\}{2\}{5\}{0\}
(b) \{2\}{5\}{7\}{2\}
(c) \{2\}{5\}{7\}{2\}{5\}{7\}{2\}{5}... (goes on forever)
(d) \{5\}{7\}{2\}{2\}{7\}{2\}{5\}{7}... (goes on forever)
(e) \{5\}{7\}{2\}{5\}{0\}

**Concept:** creating and growing lists

12. Suppose:

```c
int n[3] = {2,3,4};
Node *a = newNode(7,0);
Node *b = newNode(5,0);
Node *c = newNode(6,0);
```

Using the node and integer linked-list operations from the text, which of the following would not create a linked list holding three integers in ascending order?

(a) b->next = c; a->next = b; items = a;
(b) items = newNode(n[0],newNode(n[1],newNode(n[2],0)));
(c) items = join(n[0],join(n[1],join(n[2],0)));
(d) items = intsToList(n);
(e) items = 0;
   for (i = 0; i < 3; ++i)
      items = join(n[i],items);
13. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions adds a node \( b \) to the front of the list? After the operation, \( k \) should point to a list of three nodes.

(a) \( \text{tail}(b) = k; \)
(b) \set{T}(k,b); \)
(c) \set{T}(b,k); \( k = b; \)
(d) \set{T}(b,k); 

14. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions adds a node \( b \) to the front of the list? After the operation, \( k \) should point to a list of three nodes.

(a) \( b->\text{next} = k; k = b; \)
(b) \( k = b->\text{next}; \)
(c) \( b->\text{next} = k; \)
(d) \( k->\text{next} = b; \)

15. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions inserts a node \( b \) after the first node in the list? After the operation, \( k \) should point to a list of three nodes.

(a) \set{T}(\text{tail}(k)); \( b; \)
(b) \set{T}(k,b); \set{T}(b,\text{tail}(k));
(c) \set{T}(b,k); \set{T}(k,b);
(d) \set{T}(b,\text{tail}(k)); \set{T}(k,b);

16. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions inserts a node \( b \) after the first node in the list? After the operation, \( k \) should point to a list of three nodes.

(a) \( k->\text{next}->\text{next} = b; \)
(b) \( b->\text{next} = k->\text{next}; k->\text{next} = b; \)
(c) \( k->\text{next} = b; b->\text{next} = k->\text{next}; \)
(d) \( b->\text{next} = k; k->\text{next} = b; \)

17. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions adds a node \( b \) at the end of the list? After the operation, \( k \) should point to a list of three nodes.

(a) \( \text{tail}(\text{tail}(k)) = b; \)
(b) \set{T}(\text{tail}(\text{tail}(k)),b);
(c) \set{T}(\text{tail}(\text{tail}(k)),b);
(d) \set{T}(\text{tail}(\text{tail}(k))) = b;

18. Suppose variable \( k \) points to a linked list of two nodes. Which of the following actions adds a node \( b \) at the end of the list? After the operation, \( k \) should point to a list of three nodes.

(a) \( k->\text{next}->\text{next} = b; \)
(b) \( k->\text{next}->\text{next} = b; \)
(c) \( b = k->\text{next}->\text{next}; \)
(d) \( k->\text{next}->\text{next}->\text{next} = b; \)

**Concept:** shrinking lists

19. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the first node in the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( k = \text{tail}(\text{tail}(k)); \)
(b) \set{H}(\text{tail}(\text{tail}(k)),0); \)
(c) \set{H}(k,0); \)
(d) \( k = \text{tail}(k); \)

20. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the first node in the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( k->\text{value} = 0; \)
(b) \( k = k->\text{next}->\text{next}; \)
(c) \( k->\text{next}->\text{value} = 0; \)
(d) \( k = k->\text{next}; \)

21. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions does not remove the middle value from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( a = \text{tail}(k); b = \text{tail}(a); \text{setTail}(k,b); \)
(b) \( z = \text{tail}(\text{tail}(k)); \text{setTail}(k,z); \)
(c) \( \text{setTail}(k,\text{tail}(\text{tail}(k))); \)
(d) \( \text{setTail}(\text{tail}(k),0); \)

22. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions does not remove the middle value from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( z = k->\text{next}->\text{next}; k->\text{next} = z; \)
(b) \( k->\text{next} = k->\text{next}->\text{next}; \)
(c) \( a = k->\text{next}; b = a->\text{next}; k->\text{next} = b; \)
(d) \( k->\text{next}->\text{next} = 0; \)

23. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the middle value from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( a = \text{tail}(\text{tail}(k)); \text{setTail}(k,a); \)
(b) \( b = \text{tail}(k); k = b; \)
(c) \( \text{setTail}(k,\text{tail}(k)); \)
(d) \( c = \text{tail}(\text{tail}(k)); k = c; \)

24. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the middle value from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( b = k->\text{next}; k = b; \)
(b) \( a = k->\text{next}->\text{next}; k->\text{next} = a; \)
(c) \( c = k->\text{next}->\text{next}; k = c; \)
(d) \( k->\text{next} = k->\text{next}; \)

25. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the last node and value from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( \text{setTail}(\text{tail}(k),0); \)
(b) \( b = \text{tail}(k); \text{getNodeNext}(k,b); \)
(c) \( k = \text{setTail}(\text{tail}(\text{tail}(k)),0); \)
(d) \( k = \text{tail}(k); k = \text{tail}(k); \)

26. Suppose variable \( k \) points to a linked list of three nodes. Which of the following actions removes the last node from the list? After the operation, \( k \) should point to a list of two nodes.

(a) \( k = k->\text{next}; k->\text{value} = k->\text{next}->\text{value}; \)
(b) \( k->\text{next}->\text{next} = 0; \)
(c) \( b = k->\text{next}; k->\text{next} = b; \)
(d) \( k->\text{next}->\text{next}->\text{value} = 0; \)

**Concept:** manipulating list values

27. Suppose variable \( k \) points to a linked list of two nodes holding integer values. Which of the following does not sum the values in the list?

(a) \( \text{sum} = \text{getIndex}(k,0); \text{tail}(k); \)
(b) \( \text{sum} = \text{getIndex}(k,0); \text{sum} += \text{getIndex}(k,1); \)
(c) \( \text{sum} = \text{head}(k); \text{sum} += \text{head}(\text{tail}(k)); \)
(d) \( \text{sum} = \text{getIndex}(k,0) + \text{getIndex}(k,1); \)

28. What does the following function do? Assume \( \text{items} \) is a list and always has at least one item.
Node *g(Node *items, void *value)
{
    return join(value, tail(items));
}

(a) creates a new list with the value replacing the first value in items
(b) replaces the first value in the given list
(c) prepends a value to the given list
(d) creates a new list with a value prepended to the given list

29. What does the following function do? Assume items is a list and always has at least two items.

Node *g(Node *items, void *value)
{
    setHead(items, value);
    return items;
}

(a) creates a new list with a value prepended to the given list
(b) prepends a value to the given list
(c) replaces the first value in the given list
(d) replaces the second value in the given list

30. What does the following function do? Assume items is a list and always has at least two items.

Node *g(Node *items, void *value)
{
    setHead(tail(items), value);
    return items;
}

(a) prepends a value to the given list
(b) creates a new list with a value prepended to the given list
(c) replaces the first value in the given list
(d) replaces the second value in the given list

Concept: walking a list

31. What does the following loop do? Assume items is a list of integer values.

    t = 0;
    s = items;
    while (s != 0)
    {
        t += head(s);
        s = tail(s);
    }

(a) sums of all odd values in the list into t
(b) sums of every other value in the list into t
(c) sums of all values in the list into t
(d) sums of all even values in the list into t

32. What does the following loop do? Assume items is a list of integer values.

    t = 0;
    s = items;
    while (s != 0)
    {
        t += head(s);
        s = tail(tail(s));
    }

(a) sums of all even values in the list into t
(b) sums of all values in the list into $t$
(c) sums of every other value in the list into $t$
(d) sums of all odd values in the list into $t$

33. What is wrong with the following loop? Assume $items$ is a list of integer values.

```c
int t = 0;
int s = items;
while (s != 0)
{
    t += head(s);
    items = tail(s);
}
```

(a) the tail needs to be taken before the head is processed
(b) it can fall into an infinite loop (fossilized pattern)
(c) the program may crash if the list is empty
(d) the program may crash if the list length is not even

34. What is wrong with the following loop? Assume $items$ is a list of integer values.

```c
int t = 0;
int s = items;
while (s != 0)
{
    t += head(s);
    s = tail(tail(s));
}
```

(a) it can fall into an infinite loop (fossilized pattern)
(b) the tail needs to be taken before the head is processed
(c) the program may crash if the list length is not even
(d) the program may crash if the list is empty

35. What is wrong with the following loop? Assume $items$ is a list of integer values.

```c
int t = 0;
int s = items;
while (s != 0)
{
    s = tail(s);
    t += head(s);
}
```

(a) the function correctly sums the values in the list
(b) it can fall into an infinite loop (fossilized pattern)
(c) the program may crash if the list is empty
(d) the program may crash if the list is not empty

36. What is wrong with the following loop? Assume $items$ is a list of integer values with at least one item.

```c
int t = head(items);
int s = tail(items);
while (s != 0)
{
    t += head(s);
    s = tail(s);
}
```

(a) the function correctly sums the values in the list
(b) the program may crash if the list length is exactly one
(c) it can fall into an infinite loop (fossilized pattern)
(d) the program may crash if the list length is greater than 2

37. What is wrong with the following loop? Assume $items$ is a list of integer values.
\[ t = \text{head}(\text{items}); \]
\[ s = \text{tail}(\text{items}); \]
\[ \text{while} \ (s \neq 0) \]
\[ \{ \]
\[ t += \text{head}(s); \]
\[ s = \text{tail}(s); \]
\[ \} \]

(a) the program may crash if the list is empty
(b) the function correctly sums the values in the list
(c) the program may crash if the list is not empty
(d) it can fall into an infinite loop (fossilized pattern)

38. What does the following loop not do? Assume \textit{items} is a list.
\[ t = 0; \]
\[ s = \text{items}; \]
\[ \text{while} \ (s \neq 0) \]
\[ \{ \]
\[ t += 1; \]
\[ s = \text{tail}(s); \]
\[ \} \]

(a) stores one less than the length of the list into \( t \)
(b) increments \( t \) for every node encountered
(c) implements the counting pattern
(d) stores the length of the list into \( t \)

39. What does the following loop do? Assume \textit{items} is a list.
\[ t = 0; \]
\[ s = \text{items}; \]
\[ \text{while} \ (s \neq 0) \]
\[ \{ \]
\[ s = \text{tail}(s); \]
\[ t += 1; \]
\[ \} \]

(a) stores one less than the length of the list into \( t \)
(b) implements the filtered-count pattern
(c) stores the length of the list into \( t \)
(d) stores one more than the length of the list into \( t \)

40. What does the following loop do? Assume \textit{items} is a list and always has at least one item.
\[ t = 0; \]
\[ s = \text{items}; \]
\[ \text{while} \ (\text{tail}(s) \neq 0) \]
\[ \{ \]
\[ t += 1; \]
\[ s = \text{tail}(s); \]
\[ \} \]

(a) stores the length of the list into \( t \)
(b) stores one less than the length of the list into \( t \)
(c) stores one more than the length of the list into \( t \)
(d) implements the filtered-count pattern