1. Mark the following statements as true or False.

**a. In a linked list, the order of the elements is determined by the order in which the nodes were created to store the elements.**

**Ans**: False

**b. In a linked list, memory allocated for the nodes is sequential.**

**Ans:** False

**c. A single linked list can be traversed in either direction.**

**Ans:** False

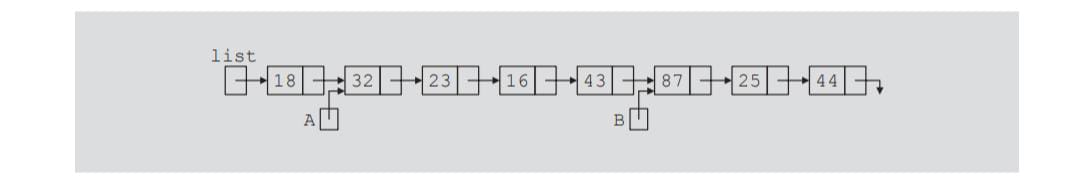
**d. In a linked list, nodes are always inserted either at the beginning or the end because a linked link is not a random access data structure.**

**Ans:** False

**e. The head pointer of a linked list cannot be used to traverse the list.**

**Ans**: False

Consider the linked list shown in Figure 5-35.



Assume that the nodes are in the usual info-link form. Use this list to answer Exercises 2 through 7. If necessary, declare additional variables. (Assume that list, p, s, A, and B are

pointers of type nodeType.)

2. What is the output of each of the following C++ statements?

**a. cout << list->info;**

Ans:18

**b. cout << A->info;**

Ans: 32

**c. cout << B->link->info;**

Ans:25

**d. cout << list->link->link->info**

Ans:23

**3. What is the value of each of the following relational expressions?**

**a. list->info >= 18**

Ans:True

**b. list->link == A**

**Ans: True**

**c. A->link->info == 16**

Ans:False

**d. B->link == NULL**

Ans:False

**e. list->info == 18**

Ans:True

**4. Mark each of the following statements as valid or invalid. If a statement is**

**invalid, explain why.**

**a. A = B;**

Ans: Invalid

Because A is pointing towards a different pointer(18) and B is pointing towards a different pointer(87)

**b. list->link = A->link;**

Ans: Invalid

Because list->link is pointing towards value 32 and A->link is pointing towards 23

**c. list->link->info = 45;**

Ans: Invalid

Because **list->link->info** is pointing towards value 32

**d. \*list = B;**

Ans: The statement is valid

**e. \*A = \*B;**

Ans: Invalid

This statement is invalid. Because the address of A store in and the address of B stores in 5 node.

**f. B = A->link->info;**

Ans: Invalid

Because B containes value which is 87 and **B = A->link->info** contains value which is 16

**g. A->info = B->info;**

Ans: Invalid Because

A->info =32

B->info = 87

**h. list = B->link->link;**

Ans: Invalid

Because list node contains 87 value while **list = B->link->link** containes null value

**I. B = B->link->link->link;**

Ans: Invalid

Because **B** containes value 87 and **B->link->link->link** containes Null value

**5. Write C++ statements to do the following:**

**a. Make A point to the node containing info 23.**

Ans: A = A->link

**b. Make list point to the node containing 16.**

Ans: list = A->link->link

**c. Make B point to the last node in the list.**

Ans: B = B->link->link

**d. Make list point to an empty list.**

Ans: List = NULL;

**e. Set the value of the node containing 25 to 35.**

Ans:

Node \* newNode = new Node();

newNode->info = 35;

B->info = newNode;

newNode->link = Rptr;

**f. Create and insert the node with info 10 after the node pointed to by A.**

Node \* newNode = new Node();

newNode->info = 10;

while(list->info!=23){

Prev= list;

list= list->link;

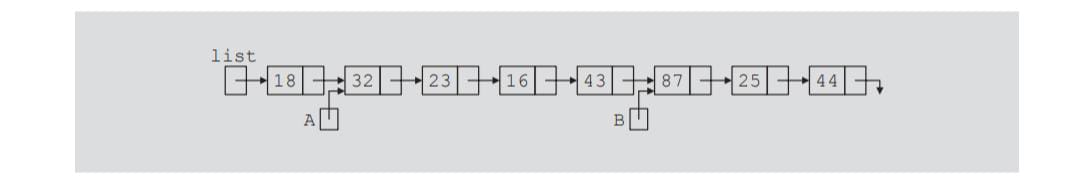
}

Prev->link = newNode;

newNode->link = list;

**g. Delete the node with info 23. Also, deallocate the memory occupied**

**by this node.**



Ans :

while(list->link->info!=23){

List = list->link;

}

List->link = list->link->link;

**6. What is the output of the following C++ code?**

Ans:

18 32 23 16 43 87 25 44

**7. If the following C++ code is valid, show the output. If it is invalid, explain**

**why.**

**a. s = A;**

**p = B;**

**s->info = B;**

**p = p->link;**

**cout << s->info << " " << p->info << endl;**

Ans:

87 25

P=A

**b. p = A;**

**p = p->link;**

**s = p;**

**p->link = NULL;**

**s = s->link;**

**cout << p->info << " " << s->info << endl;**

Ans:

The code is invalid.Because after line 4 executes, the P becomes null so the s becomes null That’s why output can’t be shown.

**8. Show what is produced by the following C++ code. Assume the node is in**

**the usual info-link form with the info of type int. (list and ptr are**

**pointers of type nodeType.)**

**a.**

**Output:**

10 17 13

**b.**

**output:**

30

42

20

28

**9. Consider the following C++ statements. (The class unorderedLinkedList**

**is as defined in this chapter.)**

**unorderedLinkedList<int> list;**

**list.insertFirst(15);**

**list.insertLast(28);**

**list.insertFirst(30);**

**list.insertFirst(2);**

**list.insertLast(45);**

**list.insertFirst(38);**

**list.insertLast(25);**

**list.deleteNode(30);**

**list.insertFirst(18);**

**list.deleteNode(28);**

**list.deleteNode(12);**

**list.print();**

**What is the output of this program segment?**

Ans:18 38 2 15 45 25

**10. Suppose the input is:**

**18 30 4 32 45 36 78 19 48 75 -999**

**What is the output of the following C++ code? (The class unorderedLinkedList**

**is as defined in this chapter.)**

unorderedLinkedList<int> list;

unorderedLinkedList<int> copyList;

int num;

cin >> num;

while (num != -999)

{

if (num % 5 == 0 || num % 5 == 3)

list.insertFirst(num);

else

list.insertLast(num);

cin >> num;

}

list.print();

cout << endl;

copyList = list;

copyList.deleteNode(78);

copyList.deleteNode(35);

cout << "Copy List = ";

copyList.print();

cout << endl;

Ans:

Copy list =75

48

45

39

16

4

32

36

19

**12 . Suppose that intList is a list container and intList = {3, 23, 23, 43, 56, 11, 11, 23, 25}**

**Show intList after the following statement executes: intList.unique();**

**Ans:** The output will be

intList = {3, 23, 43, 56, 11, 25}

**13 - Suppose that intList1 and intList2 are list containers and intList1 = {3, 58, 78, 85, 6, 15, 93, 98, 25} intList2 = {5, 24, 16, 11, 60, 9} Show intList1 after the following statement executes: intList1.splice(intList1.begin(), intList2);**

**Ans:**

Int list1 = {3,58,78,86,6,15,93,98,25,5,24,16,11,60,9}

**14. What is the output of the following program segment?**

list<int> intList;

ostream\_iterator<int> screen(cout, " ");

list<int>::iterator listIt;

intList.push\_back(5);

intList.push\_front(23);

intList.push\_front(45);

intList.pop\_back();

intList.push\_back(35);

intList.push\_front(0);

intList.push\_back(50);

intList.push\_front(34);

copy(intList.begin(), intList.end(), screen);

cout << endl;

listIt = intList.begin();

intList.insert(listIt,76);

++listIt;

++listIt;

intList.insert(listIt,38);

intList.pop\_back();

++listIt;

++listIt;

intList.erase(listIt);

intList.push\_front(2 \* intList.back());

intList.push\_back(3 \* intList.front());

copy(intList.begin(), intList.end(), screen);

cout << endl;

**Output:**

**70 76 34 0 38 45 23 0 35 210**