Instructions: Opscan forms will be passed out in class. Turn in your completed opscan at class on Tuesday Oct 29. No late Opscans will be accepted.

For questions 1 through 8, assume the following singly-linked node class and variable declarations:

```
class SNode {
public:
   int   Element;
   SNode *Next;

   SNode() {Element = 0; Next = NULL; }
   SNode(const Item& E, SNode* N = NULL) {Element = E; Next = N;}
};

SNode  First(78, NULL);
SNode* Head = &First;
```

1. What is the data type of the expression `Head` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

2. What is the data type of the expression `*Head` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

3. What is the data type of the expression `Head->Next` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

4. Assuming that additional `SNode` objectss were created and linked so that the expression is logically valid, what would be the data type of the expression `Head->Next->Next->Element` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

5. Assuming that additional `SNode` objects were created and linked so that the expression is logically valid, what would be the data type of the expression `ptr->Next->Element->Next` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

6. What is the data type of the expression `Head.Element` ?

   1) `*SNode`
   2) `SNode*`
   3) `int`
   4) `&SNode`
   5) `SNode`
   6) None of these

7. Which of the following is of type `SNode`?

   1) `&Head`
   2) `Head->Next`
   3) `(*Head).Next`
   4) `*(Head->Next)`
   5) All of them
   6) 2, 3 and 4 only
   7) 2 and 4 only
   8) 3 and 4 only
   9) None of these

8. Assuming that the target of `Head` is an `SNode` contained in a linked list, which of the following statements advances `Head` to point to the next node of that list (assuming that there is one)?

1) `Head++;`
2) `Head = Head->Next;`
3) `Head = (*Head).Next;`
4) `Head = *Head;`
5) `Head->Next = Head;`
6) 1 and 2 only
7) 1 and 3 only
8) 2 and 3 only
9) None of these

For questions 9 through 11, assume the `SNode` declaration given above, and the following code fragment:

```
SNode* p = new SNode(12);
SNode* q = new SNode(5);
q->Next  = p;
```

Hint: draw the resulting memory layout.

9. Which of the following expressions has the value 5?

1) `q`
2) `q->Element`
3) `q->Next->Element`
4) `q->Next`
5) `p`
6) `p->Element`
7) `p->Next->Element`
8) `p->Next`
9) None of these

10. Which of the following expressions has the value 12?

1) `q->Next`
2) `q->Next->Element`
3) `p->Element`
4) `p->Next->Element`
5) 1 and 3 only
6) 2 and 3 only
7) 1 and 4 only
8) 2 and 4 only
9) None of these

11. Which of the following expressions has the value `NULL`?

1) `p`
2) `q`
3) `q->Next`
4) `q->Next->Next`
5) None of these

---

12. Again assume the `SNode` declaration above, and the following code, which builds a linked list with the numbers 18 and 32 as its components, has a missing statement. What should that statement be?

```
SNode* p;
SNode* q;

p = new SNode(18, NULL);
q = new SNode(32, NULL);
_____________________  // <-- Statement is missing here
```

1) `p = q;`
2) `p->Next = new SNode;`
3) `p->Next = q;`
4) `p->Next = q->Next;`
5) `q = p->Next;`
6) None of these

For questions 13 and 14, assume the `SNode` declaration given above, and the declaration: `SNode* Head;`

Also assume that `Head` is the head pointer to a linked list of many `SNodes`.

13. Which statement renders the second node in the list inaccessible, and only the second node?

1) `Head = Head->Next;`
2) `Head = Head->Next->Next;`
3) `Head->Next = Head;`
4) `Head->Next = Head->Next->Next;`
5) 1 and 3 only
6) 2 and 3 only
7) 2 and 4 only
8) None of these

14. Which statement changes the data element of the second node in the list?

1) `Head->Element = 42;`
2) `Head->Next->Element = 42;`
3) `*(Head->Next).Element = 42;`
4) `*(headPtr->Next) = 42;`
5) 1 and 3 only
6) 2 and 3 only
7) 2 and 4 only
8) None of these

For questions 15 through 17, assume the earlier declaration of the class `SNode` and the declarations below:

```
SNode* Head;
SNode* Current;
const int initVal = 0;
```

15. If `Head` points to the first node in a properly constructed linked list of many nodes, which code segment below stores the value `initVal` in each node?

1)
```
Current = Head;
while (Current != NULL) {
   Current->Element = initVal;
   Current = Current->Next;
}
```

2)
```
Current = Head;
while (Current != NULL) {
   Current->Element = initVal;
   Current = (*Current).Next;
}
```

3)
```
Head->Element = initVal;
Current = Head;
while (Current->Next != NULL) {
   Current->Next->Element =
      Current->Element;
   Current = Current->Next;
}
```

4)
```
Current = Head;
while (Current != NULL) {
   Current->Element = initVal;
   Current++;
}
```

5)
```
Current = Head;
while (Current != NULL) {
   Current->Element =
      initVal;
   Current = Next;
}
```

6) 1 through 5 will all work

7) 1 through 4 will all work, but not 5

8) 1 and 2 only

9) 1, 2 and 3 only

10) None of these

16. Which, if any, of the code fragments given in the previous question would work correctly only if the list were nonempty? (Use the same choices for your answers.)

17. Which of the following code segments outputs the data in the list, visiting each node from first to last (assuming the list is not empty)?

1)
```
Current = Head;
while (Current != NULL) {
    cout << Current->Element
         << endl;
    Current = Current->Next;
}
```

2)
```
Current = Head;
while (Current->Next != NULL) {
    cout << Current->Element
         << endl;
    Current = Current->Next;
}
```

3)
```
Current = Head;
do {
    cout << Current->Element
         << endl;
    Current = Current->Next;
} while (Current != NULL);
```

4)
```
Current = Head->Next;
while (Current != NULL) {
    cout << Current->Element
         << endl;
    Current = Current->Next;
}
```

5) All of them

6) All except 2

7) 1 and 3 only

8) None of these

---

18. Given the earlier declaration of the class `SNode` and the declarations:

```
SNode* Previous;
SNode* Temp;
```

Assume that the target of `Previous` is an `SNode` contained in a linked list. Which of the following code segments removes (and deallocates) the node that follows the target of `Previous` without breaking or corrupting the list? You may assume that there is a node following the target of `Previous`, and that that node also has a successor.

1)
```
Temp = Previous;
Previous->Next = Temp->Next;
delete Temp;
```

2)
```
Temp = Previous->Next;
Previous = Temp->Next;
delete Temp;
```

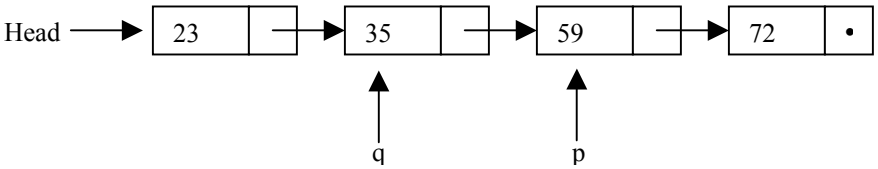
3)
```
Temp = Previous->Next;
Previous->Next = Temp->Next;
delete Temp;
```

4)
```
Temp = Previous->Next;
Previous = Temp->Next;
delete Previous;
```

5)
```
Temp = Previous->Next;
Previous = Temp->Next;
delete Previous->Next;
```

6) None of these

For questions 19 through 24, assume that `Head`, `p` and `q` have all been declared as `SNode*` and that the following list structure has been created. (Use the structure below as your starting point for each question.)



Determine what the execution of the given code fragment would do to the given structure. Choose your answers from those given on the following page.

19.
```
p->Next = q;
```

20.
```
q->Next = p->Next;
p->Next = NULL;
```

21.
```
q->Next = p->Next;
q->Next->Next = p;
p->Next = NULL;
```

22.
```
q->Next = NULL;
delete p;
```

23.
```
delete (p->Next);
```

24.
```
q->Next = Head;
delete p;
```

---

25. Suppose that the following destructor were implemented for the class `SNode`:

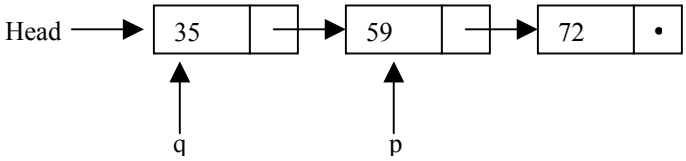
```
SNode::~SNode() {

    delete Next;
}
```

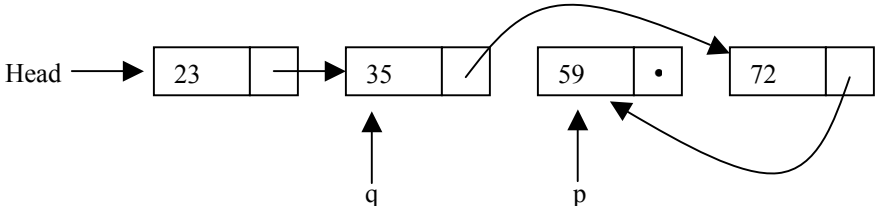
Given the list structure shown above, what would result after executing: `delete p;`

1) A correctly structured list with two nodes.
2) A correctly structured list with two nodes, and one disconnected node.
3) A list with two nodes, incorrectly terminated.
4) A list with two nodes, incorrectly terminated, and one disconnected node.
5) None of these

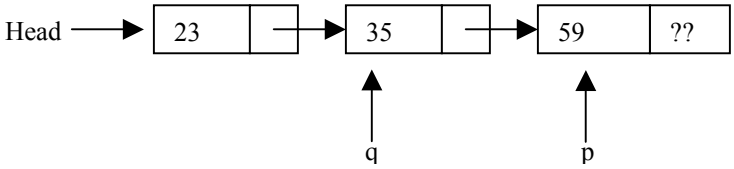
Choose from the following answers. Question marks (??) indicate the pointer has either an unknown or an invalid, value.

1) 

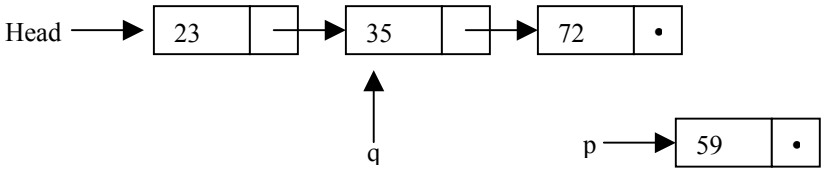
---

2) 

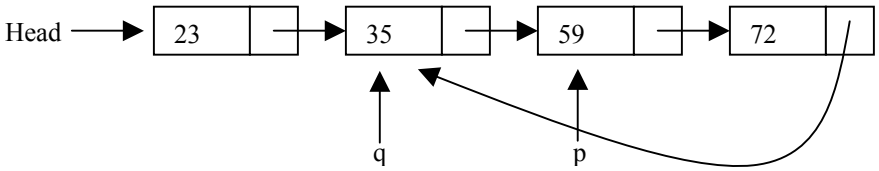
---

3) 

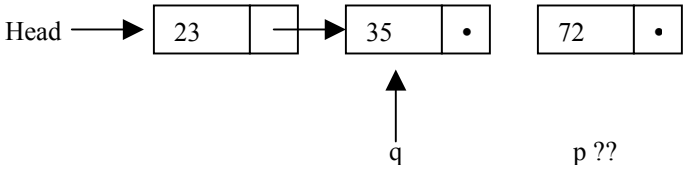
---

4) 

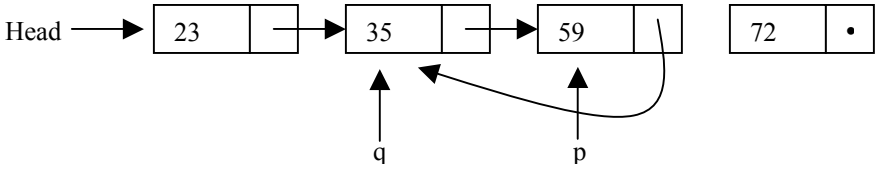
---

5) 

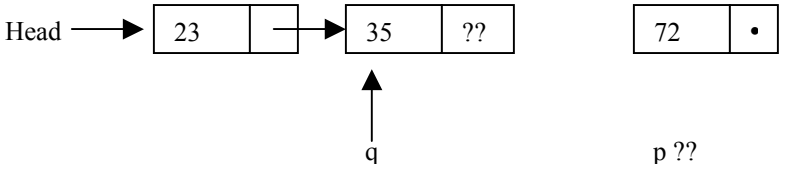
---

6) 

---

7) 

---

8) 

9) The given code contains a syntax error.

10) None of these