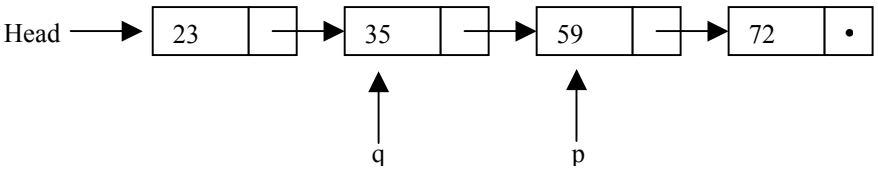
Instructions: This homework assignment covers some basics of linked lists in C++. The answers to these questions can be determined from the lecture notes and the assigned readings in the text and Parlante.

Opscan forms will be passed out in class. Write your name and code your ID number on the Opscan form. Turn in your completed Opscan at class on the day specified below. No late Opscans will be accepted.

For questions 1 through 5 determine what the execution of the given code fragment would do, assuming the following list structure as your starting point (for each question):




```
struct ListNode {
   int       Element;
   ListNode* Next;
};

ListNode *Head, *p, *q, *t;
```

Match each code fragment to one of the choices given on page 2.

1.
```
*(q->Next) = *p;
delete q->Next;
```

2.
```
p = Head;
Head = p->Next;
delete p;
p = q->Next;
```

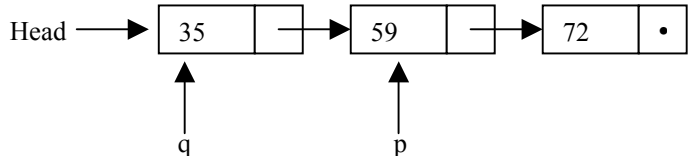
3.
```
*q = *p;
q->Element = p->Element - 24;
p->Next = q->Next->Next;
```

4.
```
delete (*q->Next);
```

5.
```
Head->Next = NULL;
t = Head;
Head = p->Next;
Head->Next = p;
p->Next = q;
q->Next = t;
t = NULL;
```

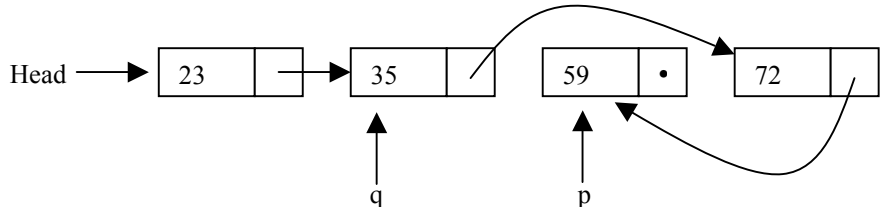
Choose from the following answers. Question marks (??) indicate the pointer has an unknown, or invalid, value.

1)



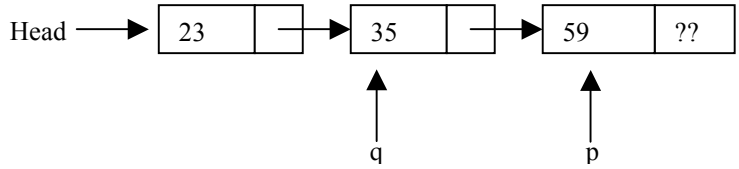
---

2)



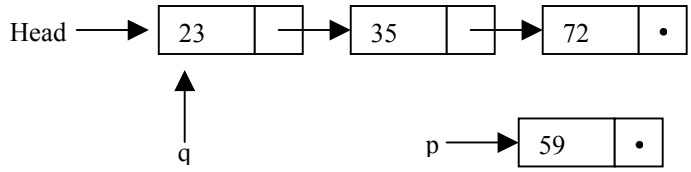
---

3)



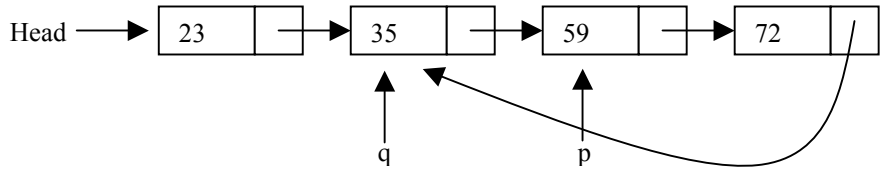
---

4)



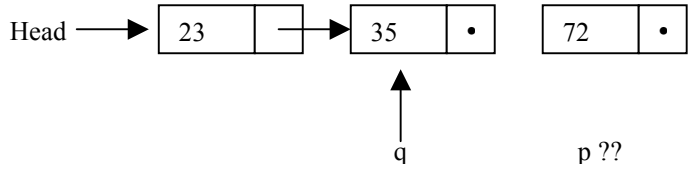
---

5)



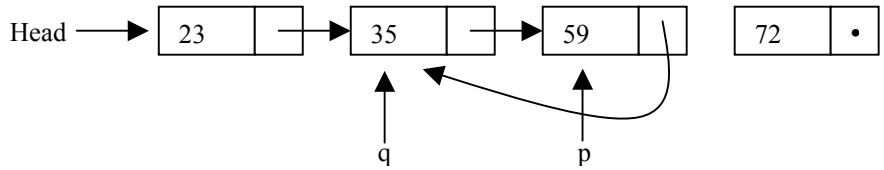
---

6)



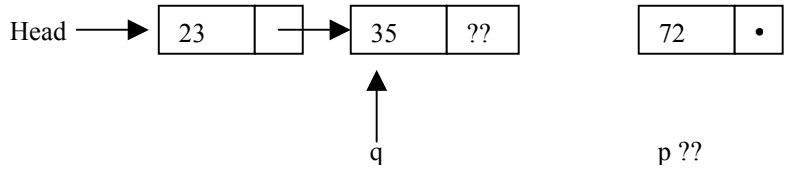
---

7)



---

8)



9) The given code contains a syntax error.

10) None of these

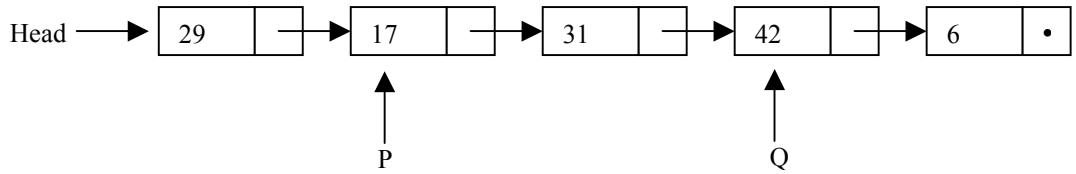
Due Friday July 26 – In class

For questions 6 through 10, assume the class `LinkNode` from the course notes, plus the class `ItemType` and pointers defined below:

```
class ItemType {
private:
   int Value;
public:
   ItemType() {Value = 0;}
   ItemType(int newValue) {Value = newValue;}
   void setValue(int newValue) {Value = newValue;}
   int  getValue() const {return Value;}
};

LinkNode *Head, *P, *Q, *T=NULL;
```

Assume the following initial list structure for each question:



Determine what the execution of the given code fragment would do, assuming the initial list structure given above as your starting point for each question. Match each code fragment to one of the choices given on page 4.

6. Head = `Q->getNext();`

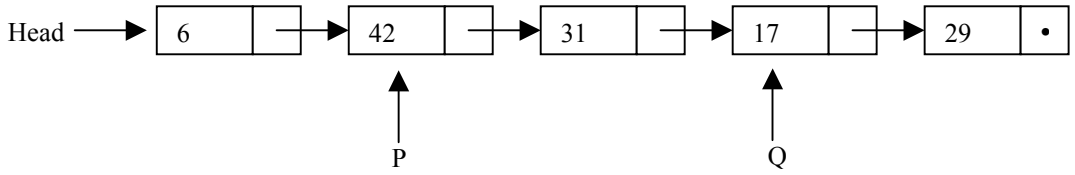
7.
```
P->setData(Head->getData());
Head->setData(Q->getData().getValue()-25);
```

8.
```
Head->setNext(P);
P->getNext()->setNext(Q);
Q->setData(P->getNext()->getNext()->getData());
```

9.
```
Head->setNext(Q->getNext()->getNext());
T = Q->getNext();
Q->getNext()->setNext(Q);
Q->setNext(P->getNext());
P->getNext()->setNext(P);
P->setNext(Head);
Head = T;
T = NULL;
```

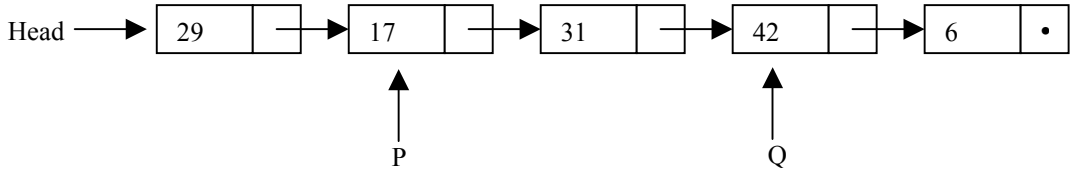
10.
```
delete Head;
Head = Q->getNext();
```

1)



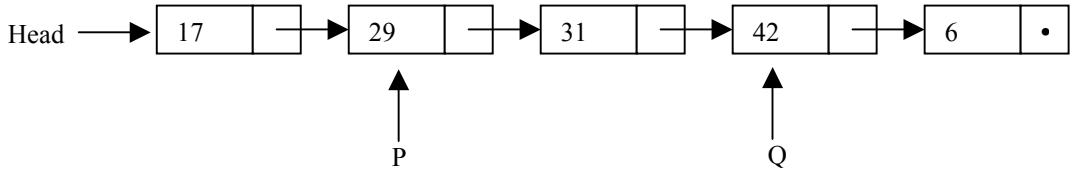
---

2)



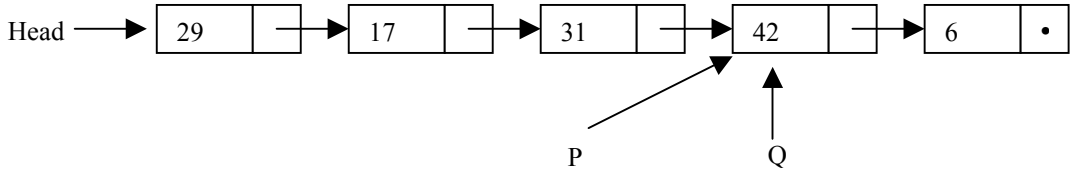
---

3)



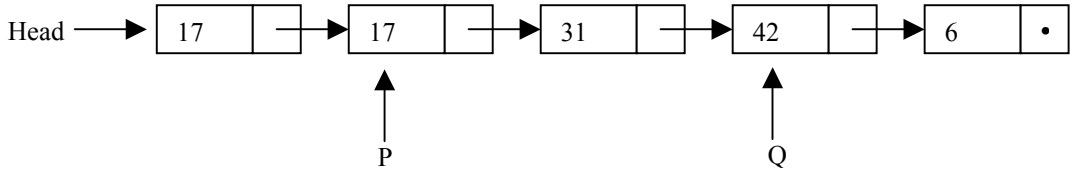
---

4)



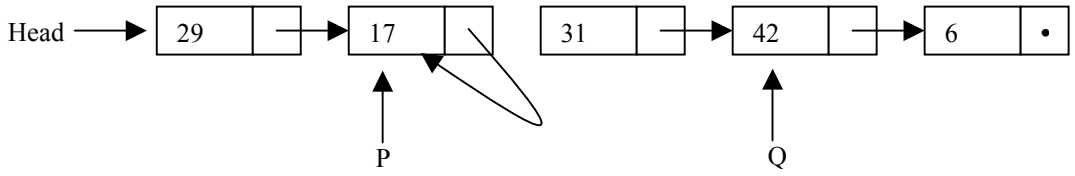
---

5)



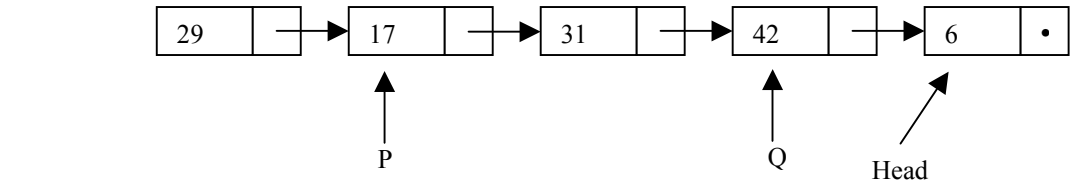
---

6)



---

7)



---

8) The given code contains a syntax error.

9) None of these.

Due Friday July 26 – In class

For questions 11 through 15, assume the classes `LinkNode` and `LinkList` from the course notes (with the necessary logical fixes in `LinkList`), and the class `Contact` defined below:

```
class Contact {
private:
   string Name;
   string Phone;
public:
   Contact();
   Contact(string initName,
           string initPhone);
   string getName() const;
   string getPhone() const;
   void   setName(string newName);
   void   setPhone(string newPhone);
   bool operator==(const Contact&
                         Other);
   void printContact(ostream& Out);
};

Contact::Contact() {
   Name  = "Anonymous";
   Phone = "BR-549";
}

Contact::Contact(string initName,
                 string initPhone) {
   Name  = initName;
   Phone = initPhone;
}
```

```
string Contact::getName() const {
   return Name;
}

string Contact::getPhone() const {
   return Phone;
}

void Contact::setName(string newName) {
   Name = newName;
}

void Contact::setPhone(string newPhone) {
   Phone = newPhone;
}

bool Contact::operator==(const Contact&
                                   Other) {
   return ( (Name == Other.Name) &&
            (Phone == Other.Phone) );
}

void Contact::printContact(ostream& Out)
{

   Out << Name
       << setw(30 - Name.length()) << ' '
       << Phone
       << endl;
}
```

Assume that `Contact` has been `typedef`'d to be synonymous with `ItemType`. Consider the problem of implementing a function (not a member function of any of the classes involved) to search a `LinkList` to update the phone number for a given `Contact` object:

```
void UpdatePhone(LinkList& L, Contact C, string NewNum) {   // header
   _____________________________________________________________________;
                                                          // line 1
   while ( ___________________________________________________________ ) {
                                                          // line 2
      if (L.getCurrentData() == C)  {
                                                          // line 3
         L.getCurrentData().setPhone(NewNum);                          //
line 4
         return;
                                                          // line 5
                          } //if
      _____________________________________________________________________;
                                                          // line 6
   } //while
```

```
         return;
                                  // line 7
      } //UpdatePhone
```

11. How should be blank in line 1 be filled?

1) `L.Advance()`
2) `L.gotoHead()`
3) `L.gotoTail()`
4) `ItemType Temp = L.getCurrentData();`
5) None of these

12. How should be blank in line 2 be filled?

1) `Curr != NULL`
2) `!L.isEmpty()`
3) `Curr != Tail`
4) `L.moreList();`
5) Either 1 or 4
6) None of these

13. How should be blank in line 6 be filled?

1) `Curr = Curr->Next`
2) `Curr = Curr->getNext()`
3) `L.Advance()`
4) `Curr++`
5) Either 2 or 3
6) None of these

14. If the last parameter, `NewNum`, were omitted and the `C` contact parameter actually contained the new "updated" phone number would the function still work correctly?

1) Yes 2) No 3) Maybe

15. Is Boolean Short-Circuiting necessary in the compiler for the above function to work correctly?

1) Yes
2) No
3) Maybe

For questions 16 through 20, consider a C++ program composed of two `cpp` files and two corresponding header files, as shown below. All function calls are shown, as are all `include` directives, but type declarations and function prototypes are not. In the source files, there should be only one physical occurrence of a function prototype, and one physical occurrence of a type declaration. Preprocessor directives for separate compilation, (#ifndef #define #endif), are not shown, but you should assume they are used wherever needed.

```
// main.h
. . .
```

```
// main.cpp
#include "Foo.h"
. . .
int main() {
   Widget R;  // struct type Widget
   Foo(R);
   . . .
}
```

```
// Foo.h
#include "main.h"
. . .
```

```
// Foo.cpp
#include "Foo.h"
. . .
void Foo(Widget& Data) {
. . .
   Data = Bar();
. . .
}

Widget Bar() {
. . .
}
```

One of our goals is for the two `cpp` files to be separately compilable. That is, we want to be able to compile `main.cpp` and `Foo.cpp` independently of each other. We also have the goal that no identifier declaration should be made available in a place where it is not needed.

Choose from the following answers:

1) `main.h`
2) `main.cpp`
3) `Foo.cpp`
4) `Foo.h`
5) Any of these
6) `main.h` or `main.cpp`
7) `Foo.cpp` or `Foo.h`
8) `main.h`, `Foo.h` or `Foo.cpp`
9) `main.h`, `main.cpp` or `Foo.h`
10) None of these

16. In order for the file `main.cpp` to compile separately, where **could** the prototype for the function `Foo()` be placed?

17. In order for the file `main.cpp` to compile separately, and for the most sensible organization of the code, where **should** the prototype for the function `Foo()` be placed?

18. In order for the file `Foo.cpp` to compile separately, where **could** the prototype for the function `Bar()` be placed?

19. In order for the file `Foo.cpp` to compile separately, and for the most sensible organization of the code, where **should** the prototype for the function `Bar()` be placed?

20. In order for both of the files `main.cpp` and `Foo.cpp` to compile separately, and for the most sensible organization of the code, where **should** the declaration for the type `Widget` be placed?