

Instructions: This homework assignment covers some of the basic C++ background you should have in order to take this course. The answers may be determined from any good C++ reference and a little thought and experimentation. You will submit your answers to the Curator System ([www.cs.vt.edu/curator](http://www.cs.vt.edu/curator)) under the heading Quiz1: Greatest Hits of C++.

Students who cannot achieve 80% or better on this assignment have serious deficiencies to enter CS 2604. Those students must expect to do considerable extra work in order to succeed.

For questions 1 through 5, consider the following implementation of a function to find **and return** the maximum value in an array of integers. The `for` loop is required to be implemented using pointers to access elements rather than direct array indexing.

```
int maxEntry(const int* const Data, int Sz) {           // Line 1
    if ( Data == NULL || Sz <= 0 ) return INT_MIN;    //      2
    int Count = 0;                                    //      3
    // Set hiSoFar to point to the first array element:
    const int *hiSoFar = _____;                 //      4
    // Set Current to point to the second array element:
    const int *Current = _____;                 //      5
    for ( ; Count < Sz; _____ ) {               //      6
        if ( _____ )                            //      7
            hiSoFar = Current;                        //      8
    }
    return ( _____ );                            //      9
}
```

1. How should the blank in Line 4 be filled?

- |          |                |                  |
|----------|----------------|------------------|
| 1) Data  | 4) &Data[0]    | 7) 3 or 4 only   |
| 2) *Data | 5) Data[0]     | 8) None of these |
| 3) &Data | 6) 3 or 5 only |                  |

2. How should the blank in Line 5 be filled?

- |              |                |                  |
|--------------|----------------|------------------|
| 1) hiSoFar   | 4) &Data[1]    | 7) 2 or 5 only   |
| 2) hiSoFar++ | 5) Data[1]     | 8) None of these |
| 3) Data++    | 6) 2 or 4 only |                  |

3. How should the blank in Line 6 be filled?

- |                       |                             |
|-----------------------|-----------------------------|
| 1) Count++            | 4) It should be left blank. |
| 2) Current++          | 5) None of these            |
| 3) Count++, Current++ |                             |

4. How should the blank in Line 7 be filled?

- |  |  |
|--|--|
| 1) <code>Current &gt; hiSoFar</code>           | 4) <code>*Current &lt; *hiSoFar</code> |
| 2) <code>&amp;Current &gt; &amp;hiSoFar</code> | 5) None of these                       |
| 3) <code>*Current &gt; *hiSoFar</code>         |  |

5. How should the blank in Line 9 be filled?

- |                              |                             |
|------------------------------|-----------------------------|
| 1) <code>*hiSoFar</code>     | 4) It should be left blank. |
| 2) <code>&amp;hiSoFar</code> | 5) None of these            |
| 3) <code>hiSoFar</code>      |                             |

For questions 6 through 8, assume that P and Q are pointers of the same type, and that each has been assigned a value.

6. What comparison would **determine be true iff** whether P and Q have targets with the same value?

- |                                  |                 |                  |
|----------------------------------|-----------------|------------------|
| 1) <code>&amp;P == &amp;Q</code> | 4) All of them  | 7) 2 and 3 only  |
| 2) <code>*P == *Q</code>         | 5) 1 and 2 only | 8) None of these |
| 3) <code>P == Q</code>           | 6) 1 and 3 only |                  |

7. What comparison would **determine be true iff** whether P and Q have the same target?

- |                                  |                 |                  |
|----------------------------------|-----------------|------------------|
| 1) <code>&amp;P == &amp;Q</code> | 4) All of them  | 7) 2 and 3 only  |
| 2) <code>*P == *Q</code>         | 5) 1 and 2 only | 8) None of these |
| 3) <code>P == Q</code>           | 6) 1 and 3 only |                  |

8. What comparison would **determine be true iff** whether P and Q store the same value?

- |                                  |                 |                  |
|----------------------------------|-----------------|------------------|
| 1) <code>&amp;P == &amp;Q</code> | 4) All of them  | 7) 2 and 3 only  |
| 2) <code>*P == *Q</code>         | 5) 1 and 2 only | 8) None of these |
| 3) <code>P == Q</code>           | 6) 1 and 3 only |                  |

9. Assume the variable declarations:

```
int Foo = 0;
int *ptr = &Foo;
```

Which of the following statements will change the value of `Foo` to 1?

- |                           |                 |                   |
|---------------------------|-----------------|-------------------|
| 1) <code>ptr++;</code>    | 5) All of these | 9) 3 and 4 only   |
| 2) <code>Foo++;</code>    | 6) 1 and 2 only | 10) None of these |
| 3) <code>(*Foo)++;</code> | 7) 1 and 4 only |                   |
| 4) <code>(*ptr)++;</code> | 8) 2 and 4 only |                   |

10. Both code fragments below will compile but the one on the right will (on most systems) cause a runtime error. Why?

```
int x = 5;
int *p = new int(x);
delete p;
```

```
int x = 5;
int *p = &x;
delete p;
```

- |  |   |
|--|---|
| 1) Assigns an address to an <code>int</code> variable. | 3) Deletes a statically allocated variable. |
| 2) Assigns an <code>int</code> variable to a pointer.  | 4) None of these                            |

For questions 11 through 14 assume that we have a dynamically allocated array A of integers of dimension Size, with memory layout as shown:

```
const int Size = 5;
int *A = new int[Size];
```

Index	Address
0	0x007D0E70
1	0x007D0E74
2	0x007D0E78
3	0x007D0E7C
4	0x007D0E80

11. Which code fragment(s) could be inserted in the blank in order to safely initialize each element of A to zero?

```
int* p = &A[0];
for (int Idx = 0; Idx < Size; Idx++, p++) {
    _____;
}
```

- |                |                     |                   |
|----------------|---------------------|-------------------|
| 1) *A = 0;     | 4) *Idx = 0;        | 7) 1 and 3 only   |
| 2) A[Idx] = 0; | 5) All of the above | 8) 1 and 4 only   |
| 3) *p = 0;     | 6) 1 and 2 only     | 9) 2 and 3 only   |
|                |                     | 10) None of these |

12. What value will be printed by the code fragment:

```
for (int Idx = 0; Idx < Size; Idx++) {
    A[Idx] = int(&A[Idx]); // typecast converts address to int
}
cout << hex << A[3] << endl; // manipulator causes hex output
```

- |               |               |                  |
|---------------|---------------|------------------|
| 1) 0x007D0E70 | 4) 0x007D0E7C | 7) None of these |
| 2) 0x007D0E74 | 5) 0x007D0E80 |                  |
| 3) 0x007D0E78 | 6) Unknown    |                  |

13. Assuming only the initial declarations given above, what logical error(s) would result if the following statement were executed: A = new int[2\*Size];

- 1) A dangling pointer would result (a pointer whose value is the address of memory that the program does not own).
- 2) A memory leak would result (the program would own memory that it could no longer access).
- 3) Both a dangling pointer and a memory leak would result.
- 4) Neither a dangling pointer nor a memory leak, but some other logical error would result.
- 5) No logical error would result.

14. Assuming only the initial declarations given above, and execution of the correctly completed code given in question 11, what logical error(s) would result if the following statement were executed: delete [] p;

- 1) A dangling pointer would result (a pointer whose value is the address of memory that the program does not own).
- 2) A memory leak would result (the program would own memory that it could no longer access).
- 3) Both a dangling pointer and a memory leak would result.
- 4) Neither a dangling pointer nor a memory leak, but some other logical error would result.
- 5) No logical error would result.

15. Consider implementing a function to dynamically allocate an array of integers and set all its elements to zero:

```
void ZeroIt(_____ A, const int Size) {
    A = new int[Size];
    for (int Idx = 0; Idx < Size; Idx++) {
        A[Idx] = 0;
    }
}
```

Which of the following choices for the blank preceding the formal parameter A is best?

- |                           |                            |                                  |
|---------------------------|----------------------------|----------------------------------|
| 1) <code>int*</code>      | 3) <code>const int*</code> | 5) <code>const int* const</code> |
| 2) <code>int*&amp;</code> | 4) <code>int* const</code> | 6) All of the above              |

16. Which of the following statements about C++ classes is *false*?

- |  |   |
|--|---|
| 1) Classes can have private member functions.              | 4) Classes can have public data members.          |
| 2) Classes can have public, private and protected members. | 5) Aggregate assignment is permitted for classes. |
| 3) By default, members of classes are public.              | 6) None of these, (all are true).                 |

17. Which of the following C++ built-in operations are automatically defined for `class` objects?

- |                          |                 |                  |
|--------------------------|-----------------|------------------|
| 1) <code>==</code>       | 4) 1 and 2 only | 7) All of them   |
| 2) <code>=</code>        | 5) 1 and 3 only | 8) None of these |
| 3) <code>&lt;&lt;</code> | 6) 2 and 3 only |                  |

For questions 18 through 22, consider the class declaration:

```
class Farey {
private:
    int Top,
        Bottom;

public:
    Farey();
    Farey(int T, int B);
    Farey operator+(const Farey& RHS) const;
    Farey operator-(const Farey& RHS) const;
    bool operator==(const Farey& RHS) const;
    void Display(ostream& Out) const;
};

Farey::Farey() {
    Top = Bottom = 0;
}

Farey::Farey(int T, int B) {
    Top = T;
    Bottom = B;
}
```

```

Farey Farey::operator+(const Farey& RHS) const {
    return Farey(Top + RHS.Top, Bottom + RHS.Bottom);
}

Farey Farey::operator-(const Farey& RHS) const {
    return Farey(Top - RHS.Top, Bottom - RHS.Bottom);
}

bool Farey::operator==(const Farey& RHS) const {
    return ( (Top == RHS.Top) && (Bottom == RHS.Bottom) );
}

void Farey::Display(ostream& Out) const {
    Out << Top << '/' << Bottom;
}

```

Again, assuming everything necessary is in scope, consider the following code fragment:

```

Farey A(3, 5), B(1, 4), C(2, 4), D(0, 5), E;
E = A + B; // line 1
A.Display(cout); // 2
E = A + B - C; // 3
E = 2*A; // 4

```

18. After the execution of line 1, what are the values of `E.Top` and `E.Bottom`, respectively?
- |            |            |                  |
|------------|------------|------------------|
| 1) 0 and 0 | 3) 1 and 4 | 5) Unknown       |
| 2) 3 and 5 | 4) 4 and 9 | 6) None of these |
19. What is written to the stream `cout` when line 2 is executed?
- |          |            |                  |
|----------|------------|------------------|
| 1) "3/5" | 2) Nothing | 3) None of these |
|----------|------------|------------------|
20. After the execution of line 3, what are the values of `E.Top` and `E.Bottom`, respectively?
- |            |                                 |                  |
|------------|---------------------------------|------------------|
| 1) 0 and 0 | 3) 2 and 5                      | 5) Unknown       |
| 2) 4 and 9 | 4) The statement isn't allowed. | 6) None of these |
21. After the execution of line 4, what are the values of `E.Top` and `E.Bottom`, respectively?
- |             |                                 |                  |
|-------------|---------------------------------|------------------|
| 1) 6 and 10 | 3) 3 and 10                     | 5) Unknown       |
| 2) 6 and 5  | 4) The statement isn't allowed. | 6) None of these |

Consider the following code fragment:

```
Farey X(1, 2), Y(2, 4);

if ( X + X == Y )           // line 5
    cout << "X + X == Y" << endl;
else
    cout << "X + X != Y" << endl;
```

22. When the `if` statement beginning in line 5 is executed, what is written to `cout`?

1) "X + X == Y"

2) "X + X != Y"

For questions 23 through xx, consider the following class and partial implementation:

```
class Quadratic {
private:
    double Coefficient[3];
public:
    Quadratic(double a = 0.0, double b = 0.0, double c = 0.0);
    double Evaluate(double x) const;
    Quadratic operator+(const Quadratic& RHS) const;
    void Display(ostream& Out) const;
};

Quadratic::Quadratic(double a, double b, double c) {

    Coefficient[0] = a;
    Coefficient[1] = b;
    Coefficient[2] = c;
}

double Quadratic::Evaluate(double x) const {

    return ( Coefficient[0]*x*x + Coefficient[1]*x + Coefficient[2] );
}
```

23. Given the declaration: `Quadratic F(1, 2, 3);`

What value is output by the statement: `cout << F.Evaluate(2);`

1) 6

2) 11

3) 17

4) None of these

24. Given the declaration: `Quadratic G(1);`

What value is output by the statement: `cout << G.Evaluate(2);`

1) 1

2) 4

3) Not allowed.

4) None of these

A designer wants to add an addition operation to the class `Quadratic`. Consider the partial implementation:

```

_____ Quadratic::operator+(const Quadratic& RHS) const { // line 1

    double a = Coefficient[0] + RHS.Coefficient[0];
    double b = Coefficient[1] + RHS.Coefficient[1];
    double c = Coefficient[2] + RHS.Coefficient[2];

    return _____; // line 2
}

```

25. How should the blank in line 1 be filled?

- |                           |                             |
|---------------------------|-----------------------------|
| 1) <code>void</code>      | 4) It should be left blank. |
| 2) <code>Sum</code>       | 5) None of these            |
| 3) <code>Quadratic</code> |                             |

26. How should the blank in line 2 be filled?

- |                                    |                             |
|------------------------------------|-----------------------------|
| 1) <code>Quadratic(a, b, c)</code> | 4) It should be left blank. |
| 2) <code>Sum</code>                | 5) None of these            |
| 3) <code>Quadratic(c, b, a)</code> |                             |

27. If a program will not compile which of the following must be true?

- |                                     |                     |
|-------------------------------------|---------------------|
| 1) The syntax is not correct.       | 6) 1 and 2 only     |
| 2) The style is not correct.        | 7) 1, 2, and 3 only |
| 3) The syntax is correct            | 8) 3 and 4 only     |
| 4) The design logic is not correct. | 9) None of these    |
| 5) All of the above.                |                     |

28. If a program compiles but does not produce correct output, which of the following must be true?

- |                                     |                     |
|-------------------------------------|---------------------|
| 1) The syntax is not correct.       | 6) 1 and 2 only     |
| 2) The style is not correct.        | 7) 1, 2, and 3 only |
| 3) The syntax is correct            | 8) 3 and 4 only     |
| 4) The design logic is not correct. | 9) None of these    |
| 5) All of the above.                |                     |

29. If a program compiles and produces correct output when given one sample input, which of the following must be true?

- |                                 |                     |
|---------------------------------|---------------------|
| 1) The syntax is not correct.   | 6) 1 and 2 only     |
| 2) The style is not correct.    | 7) 1, 2, and 3 only |
| 3) The syntax is correct        | 8) 3 and 4 only     |
| 4) The design logic is correct. | 9) None of these    |
| 5) All of the above.            |                     |

For questions 30 through 40, assume the following singly-linked node class:

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

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

30. Assume the following declarations:

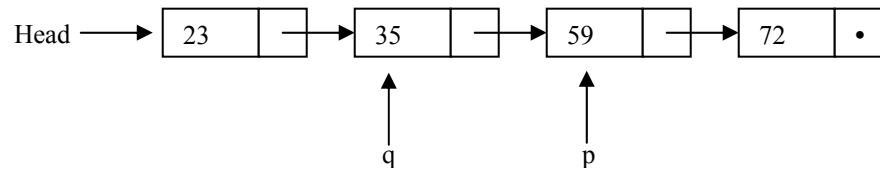
```
SNode* Head;
SNode* Current;
const int initVal = 0;
```

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?

- |   |   |
|---|---|
| <p>1) <code>Current = Head;</code><br/> <code>while (Current != NULL) {</code><br/>             <code>Current-&gt;Element = initVal;</code><br/>             <code>Current = Current-&gt;Next;</code><br/>           <code>}</code></p>   | <p>5) <code>Current = Head;</code><br/> <code>while (Current != NULL) {</code><br/>             <code>Current-&gt;Element =</code><br/>                 <code>initVal;</code><br/>             <code>Current = Next;</code><br/>           <code>}</code></p> |
| <p>2) <code>Current = Head;</code><br/> <code>while (Current != NULL) {</code><br/>             <code>Current-&gt;Element = initVal;</code><br/>             <code>Current = (*Current).Next;</code><br/>           <code>}</code></p>  | <p>6) 1 through 5 will all work</p> <p>7) 1 through 4 will all work,<br/>but not 5</p>  |
| <p>3) <code>Head-&gt;Element = initVal;</code><br/> <code>Current = Head;</code><br/> <code>while (Current-&gt;Next != NULL) {</code><br/>             <code>Current-&gt;Next-&gt;Element =</code><br/>                 <code>Current-&gt;Element;</code><br/>             <code>Current = Current-&gt;Next;</code><br/>           <code>}</code></p> | <p>8) 1 and 2 only</p> <p>9) 1, 2 and 3 only</p> <p>10) None of these</p>   |
| <p>4) <code>Current = Head;</code><br/> <code>while (Current != NULL) {</code><br/>             <code>Current-&gt;Element = initVal;</code><br/>             <code>Current++;</code><br/>           <code>}</code></p>  |   |

31. 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.)

For questions 32 through 39, 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.

32. `p->Next = q;`

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

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

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

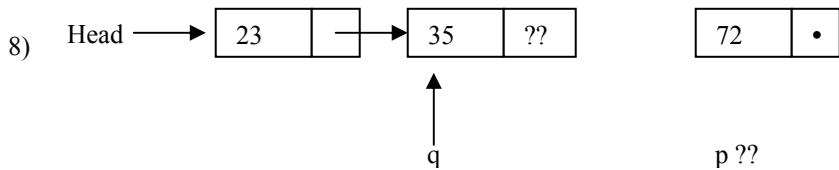
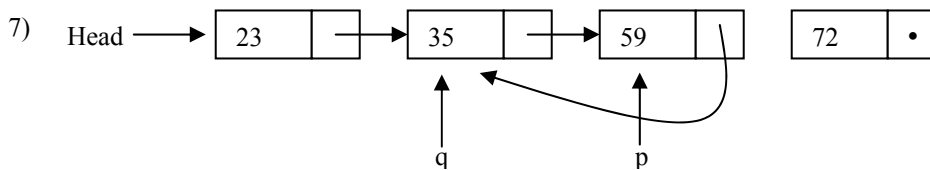
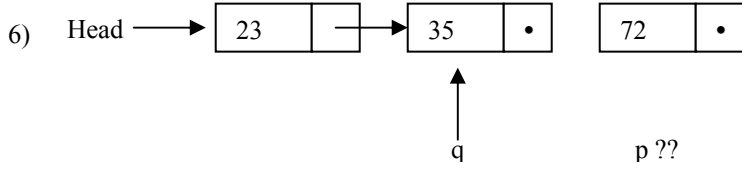
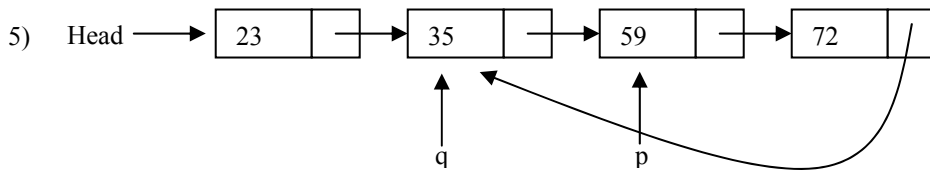
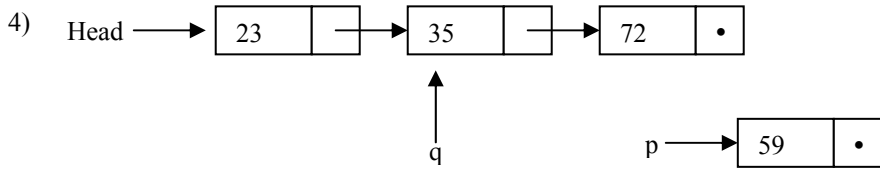
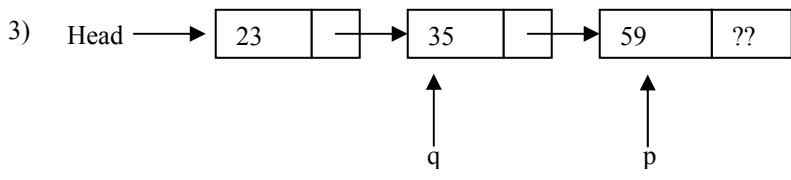
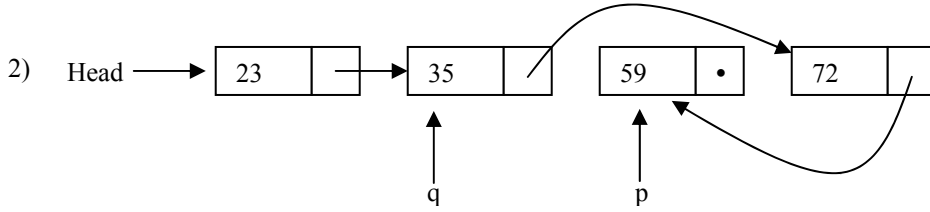
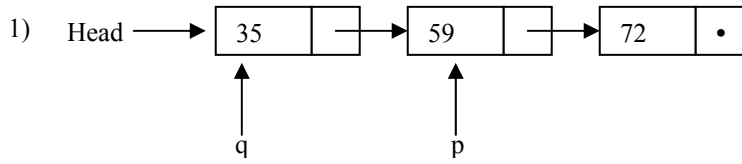
36. `delete (p->Next);`

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

38. `delete Head;`  
`Head = q;`

39. `delete p;`

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



9) The given code contains a syntax error.

10) None of these

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

```
SNode::~SNode() {  
  
    delete Next;  
}
```

Given the list structure shown before question 32, 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