This exam is OPEN Textbook and CLOSED notes. The use of computing and/or communicating devices is NOT permitted.

Do not remove any sheets from this exam book. Answer all questions in the space provided. No additional sheets are permitted.

Work independently. The value of each question is indicated. The total value of all questions is 100.

Write your name and student number in the space below. Do the same on the top of each sheet of this exam book.

Name: ____________________________________________

Student Number: __________________________________

Q1. _____/9  Q10. _____/4
Q2. _____/4  Q11. _____/4
Q3. _____/2  Q12. _____/8
Q4. _____/6  Q13. _____/5
Q5. _____/6  Q14. _____/5
Q6. _____/4  Q15. _____/4
Q7. _____/6  Q16. _____/8
Q8. _____/5  Q17. _____/4
Q9. _____/5  Q18. _____/11

Total

Page 1 of 34
Question 1. (9 marks). General.

(a) (1 mark). TRUE or FALSE. Functions that are constant member functions may call the class mutator functions.

(b) (1 mark). TRUE or FALSE. If the member variables in a base class are marked as protected, a derived class can directly access those variables.

(c) (1 mark). TRUE or FALSE. Friend functions are members of the class.

(d) (1 mark). TRUE or FALSE. Destructors are not inherited into the derived class.

(e) (1 mark). Circle one answer. All the code between

```c
#include <MYCLASS_H
and
endif
```

is ____________ if MYCLASS_H is defined.

A. skipped  
B. executed  
C. compiled  
D. debugged

(f) (1 mark). Circle one answer. Given a binary search tree, which traversal type would print the values in the nodes in sorted order?

A. Preorder  
B. Postorder  
C. Inorder  
D. None of the above

(g) (1 mark). Circle one answer. Algorithm A has a time complexity of O(n). Algorithm B has a time complexity of O(log n).

1. Algorithm B is always more efficient than algorithm A.
2. Algorithm B is never more efficient than algorithm A.
3. Algorithm A can sometimes be faster than algorithm B.
4. None of the above.
(h) (1 mark). **Circle one or more answers.** Which of the following statements about binary trees is NOT true?

A. Every binary tree has at least one node.
B. Every non-empty tree has exactly one root node.
C. Every node has at most two children.
D. Every non-root node has exactly one parent.

(i) (1 mark). **Circle one answer.** When a class that has pointer to a dynamically allocated array as one of its members, the class should also include _____________.

A. a copy constructor
B. a default constructor
C. a destructor
D. the assignment operator
E. all of the above.
F. none of the above
Question 2. (4 marks). Pointers and Arrays.

In the blank space for each part (a) – (d) below, provide the declaration and initialization of the variable \( i \) (for example: `int i = 10;`). Your initialization of \( i \) must make use of the variable \( j \), and lead to the expected output when used in the statements that follow it.

(a) `int j = 10;`

```cpp
// declare and initialize i:

cout << *i << endl;
j = 5;
cout << *i << endl;
```

(b) `int j = 10;`

```cpp
// declare and initialize i:

cout << i << endl;
j = 5;
cout << i << endl;
```

(c) `int j = 10;`

```cpp
// declare and initialize i:

cout << i[0] << endl;
j = 5;
cout << i[1] << endl;
```
(d) int j = 10;

// declare and initialize i:

cout << *(i[0]) << endl;
j = 5;
cout << *(i[1]) << endl;

Expected Output of (d)

10
5
**Question 3. (2 marks). Amy’s Dilemma.**

Amy (the same very good C++ programmer from the midterm test) designed and implemented a Mystery class so as objects of type Mystery can be created but not deleted by non-member functions. She did so by making the destructor of Mystery private, as shown below in Mystery.h.

```cpp
#include "YetAnotherMystery.h"
class Mystery {
    private:
        YetAnotherMystery* mystery_variable;
        ~Mystery();
    public:
        Mystery();
        Mystery(const Mystery & other);
        Mystery & operator=(const Mystery & rhs);
        void operateOnMystery();
        void print () const;
};
```

Thus, if a function that is non-member of the Mystery class attempts to delete a Mystery object, a compile time error is generated.

Now Amy met Shannon. Like Amy, he is also a very good C++ programmer. Shannon wrote a non-member function that has the following prototype:

```cpp
void mysterious ();
```

The function creates a Mystery object and would like to delete the object to avoid a memory leak. However, it cannot because the destructor of Mystery is private. Amy wants to allow only Shannon’s non-member function to delete Mystery objects but still prevent all other non-member functions from doing the same. She wants to do so without making Shannon’s function a member of her class.

Amy needs to add only one line to her Mystery.h. No other modifications to Mystery.h are allowed.

Write the line Amy should add.

**Answer:**

```cpp
friend void Mystery::operator delete (Mystery*); // Add this line
```

Amy (yes, the same one) went to Hollywood to write C++ code for a major movie studio. She wrote the following class definitions and implementations for *Movie* and *Chapter*. They respectively represent a movie and the chapters that make up the movie. She also wrote the functions *MovieDuration* and *main* to utilize these classes. The two classes, *MovieDuration* and *main* are all in one file and they compile with no errors.

```cpp
#include <iostream>
using namespace std;

class Chapter {
private:
    string title;
    float duration;
public:
    Chapter () {duration = 0.0;} // Constructor
    Chapter (string t, float d) {title = t; duration = d;} // Constructor
    string getTitle() const {return title;} // Getter
    float getDuration() const {return duration;} // Getter
};

class Movie {
private:
    string title;
    int numChapters;
    Chapter* chapters;
public:
    Movie (string t, int num_c) {
        title = t;
        numChapters = num_c;
        chapters = new Chapter[numChapters];
    }
    ~Movie () {delete [] chapters;} // Destructor
    int getNumChapters() const {return numChapters;} // Getter
    void addChapter (const Chapter& c, int i) {
        chapters[i] = *(new Chapter(c));
    }
    Chapter& getChapter (int i) const {return chapters[i];}
};

// Code continues on next page
```
int MovieDuration (Movie m) {
    float m_duration = 0.0;
    for (int i=0; i < m.getNumChapters(); ++i)
        m_duration = m_duration + m.getChapter(i).getDuration();
    return (m_duration);
}

int main() {
    string name = "The Adventures of an ECE244 Student";
    Movie ece244Movie(name, 13);

    // Code that creates the chapters and adds them to ece244Movie
    // Assume code will never add more chapters than numChapters

    cout << MovieDuration(ece244Movie) << endl;
    return 0;
}

However, it turns out that the code encounters a serious runtime problem when executed.

(a) (2 marks). Identify the problem using one short sentence. For example, if you think the problem is that the program divides by 0, then write: “the program divides by 0”. Similarly, if you think it dereferences a null pointer, then write: “program dereferences a null pointer”. Your answer should be this short and precise to receive marks.

Answer:

(b) (4 marks). The problem can be corrected by adding a single method in one of the classes. Write this method. Other than this method, you are not allowed to make any changes to the classes.

```cpp
{ Write method header here

Write method body here

}
Question 5. (6 marks) Amy in La La Land.

Amy (the same very good C++ programmer) liked the movie La La Land so much that she wrote two classes, LaLa and Land, defined and implemented them correctly. She now wishes to use them together, along with integers, to support the operations shown in `main` below.

```c++
int main() {
    LaLa a(1);
    Land b(2);
    int f = 3;

    a = a + b;  // a + b returns by value an object of type LaLa
    a = b + a;  // b + a returns by value an object of type Land
    a = a + f;  // a + f returns by value an object of type LaLa
    b = f + b;  // f + b returns by value an object of type Land

    return (0);
}
```

Amy must write a set of overloaded operators to make the above code work in `main`.

(a) (2 marks). How many of these operators must be non-member functions?

(b) (2 marks). How many of these operators may be member methods of class LaLa?

(c) (2 marks). How many of these operators may be member methods of class Land?

Hint: Amy says an overloaded operator= must be a class member.

There will be a Federal election in October of 2019. Amy (you guessed it, the same one) decided to run for local office. Vital to any election is vote counting. Thus, Amy would like you to design a recursive function called count to determine who got more votes without using “counters,” i.e., variables that increment by a constant amount (e.g., \( x = x + 1 \)).

Assume that votes are stored in a character array of length \( n \). An ‘I’ represents a vote for the incumbent, and a ‘C’ represents a vote for the challenger (assume only one challenger). When processing the array, the function should return the difference in votes between the incumbent and the challenger (a positive number if the incumbent has more votes, and a negative number if the challenger has more votes).

For example, calling the count function on the following array:

\[
\text{vArray} \begin{bmatrix} I & I & C & C & C & I & C & C & C \end{bmatrix}
\]

returns a vote difference of \(-3\).

Write your recursive implementation of the function count in the space below.

**Hint:** determine what the basis of recursion should be, then express the original problem in terms of one or more smaller (i.e., closer to the basis) problems.

```c
int count ( char *voteArray, int lowIndex, int highIndex ) {

// Here is how the function is called for an n-element vArray
cout << count(vArray, 0, n-1);
```

**Question 7. (6 marks). Recursion.**

Consider the two classes `ListNode` and `LinkedList` used to implement a linked list. They are not unlike the ones described in class or those you implemented in the lab. Only the class members relevant to the question are shown.

```cpp
class ListNode {
    private:
        int id;
        ListNode* next;
    public:
        ListNode* getNext() {return next;}
};

class LinkedList {
    private:
        ListNode* head;
    public:
        ListNode* getTail();
};
```

Write a *recursive* member function of `LinkedList` that returns a pointer to the last node in the list. If the list is empty it should return NULL. You may add new data members or methods to the classes.

```cpp
ListNode* LinkedList::getTail()
```
Question 8. (5 marks). Tree Traversals.

In the reverse-postorder traversal of a tree, the right subtree is first traversed (recursively in reverse-postorder), then the left subtree is traversed (also recursively in reverse-postorder), and finally the node is visited. That is, the order of the traversal is RLN. For example, the reverse-postorder traversal of the tree shown below is C E D A B.

Write a recursive function to perform the reverse-postorder traversal of a binary tree. Assume that visiting a node simply prints its data to cout. The body of the function should be very short (4-7 lines)! Longer code will be penalized.

You may assume the following declarations:

```cpp
class treenode {
    public:
        int data;
        treenode* left;
        treenode* right;
    }

treenode* root; // root of the tree

void reverse-postorder (treenode *myroot) {
    // This is how reverse-postorder is called
    reverse-postorder(root);
}
```

Consider the following class declaration of a node in a binary search tree, called BSTNode.

class BSTNode {
    public:
        int val;
        BSTNode *left;
        BSTNode *right;
};

Given a binary search tree, write a non-member function called printInRange that prints all the keys in the binary search tree that fall within a given range in sorted order.

For example, given the following binary search tree

```
          8
         / \
        3   10
       /   / \
      1   6   14
   /     /   \
  4     7    13
```

and the range 5 and 13, the function output is: 6 7 8 10 13.

Note that the range values are inclusive (i.e., a key is printed if it is equal to the given lower or upper bound). Write your answer below.

```cpp
bool printInRange(BSTNode root, int lowerBound, int upperBound) {
}
```
**Question 10. (4 marks). Destructors.**

In lab assignment 5, you were asked to implement a binary search tree consisting of objects of type TreeNode. Assume the following simplified definition of TreeNode.

```cpp
class TreeNode {
    public:
        int key;
        TreeNode *left;
        TreeNode *right;
};
```

Assume that your program creates a binary search tree with a pointer to its root node called root. Write the destructor of TreeNode so that the following statement deletes all the nodes in your tree (unlike your lab assignment).

```cpp
delete root;
```

Provide your answer below.

```cpp
TreeNode::~TreeNode() {
}
```

Consider the following class definitions and implementations, as well as a main function that uses them. All the code appears in one file and it compiles with no errors.

```cpp
#include <iostream>
using namespace std;

class firstOne {
private:
    int x;
    char* first;

public:
    firstOne();
    firstOne(const char* s);
};

class secondOne : public firstOne {
private:
    int y;
    char* second;

public:
    secondOne(const char* s);
    secondOne(int v);
};

firstOne::firstOne() {
x = 0;
    first = new char[1];
    first = '\0';
    cout << "Constructor 1 of firstOne done" << endl;
}

firstOne::firstOne(const char* s) {
x = 0;
    int size = strlen(s);
    first = new char[size+1];
    strcpy(first,s);
    cout << "Constructor 2 of firstOne done" << endl;
}

secondOne::secondOne(const char* s) {
y = 0;
    int size = strlen(s);
    second = new char[size+1];
    strcpy(second,s);
    cout << "Constructor 1 of secondOne done" << endl;
}

secondOne::secondOne(int v):firstOne("X") {
y = v;
    second = new char[1];
    second = '\0';
    cout << "Constructor 2 of secondOne done" << endl;
}

// Code continues on next page
```
```c
int main () {
    firstOne a;
    firstOne b(“G”);
    secondOne c(“N”);
    secondOne d(8);
}
```

Indicate what each statement in the above `main` function prints, using the table below. If no output is produced, write “NONE”.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>firstOne a;</code></td>
<td></td>
</tr>
<tr>
<td><code>firstOne b(“G”);</code></td>
<td></td>
</tr>
<tr>
<td><code>secondOne c(“N”);</code></td>
<td></td>
</tr>
<tr>
<td><code>secondOne d(8);</code></td>
<td></td>
</tr>
</tbody>
</table>
**Question 12. (8 marks). Inheritance.**

Consider the following base and derived classes. They compile with no errors.

```cpp
#include <iostream>
#include <string>
using namespace std;

class Dessert {
    protected:
        int price=0;
    public:
        Dessert () {cout << "we have dessert\n";}
        ~Dessert () {cout << "no more dessert\n";}
        void print() {cout << "dessert unknown\n";}
        virtual int cost() = 0;
    };

class Candy : public Dessert {
    protected:
        int weight_in_grams;
    public:
        Candy(int grams, int per_gram) {
            weight_in_grams = grams;
            price = per_gram;
            cout << "we have candy\n";
        }
        ~Candy() {cout << "no more candy\n";}
        void print() {
            cout << "Candy: " << weight_in_grams << " grams, at "
                  << price << " cents per gram\n";
        }
        virtual int cost() {return weight_in_grams*price;}
    };

class Cookies : public Dessert {
    protected:
        int num_dozens;
    public:
        Cookies(int dozens, int per_dozen) {
            num_dozens = dozens;
            price = per_dozen;
            cout << "we have cookies\n";
        }
        ~Cookies() {cout << "no more cookies\n";}
        void print() {
            cout << "Cookies: " << num_dozens << " dozens, at "
                  << price << " cents per dozen\n";
        }
        virtual int cost () {return num_dozens*price;}
    };
```
class IceCream : public Dessert {
protected:
    string flavor = "Vanilla";
    int scoops;

public:
    IceCream(string flv, int scps, int per_scoop) {
        flavor = flv;
        scoops = scps;
        price = per_scoop;
        cout << "we have " << flavor << " Ice Cream\n";
    }

    ~IceCream() {cout << "no more ice cream\n";}

    void print() {
        cout << "Ice Cream: " << scoops
            << " scoops, at " << price << " cents per scoop\n";
    }

    virtual int cost () {return scoops*price;}
};

class Sundae : public IceCream {
protected:
    int topings_cost;

public:
    Sundae(string flv, int scps, int per_scoop,
            int tps):IceCream(flv, scps, per_scoop) {
        topings_cost = tps;
        cout << " with topings\n";
    }

    ~Sundae() {cout << "no more sundae\n";}

    void print() {
        IceCream::print();
        cout << " topings cost is: " << topings_cost << endl;
    }

    virtual int cost () {return scoops*price + topings_cost;}
};

Consider each of the following main functions that use the above classes. You may assume that each main function includes the code of the classes above, including the #include’s and the using namespace std;

For each main function, indicate if the function compiles with no errors or not (ignore warnings). If the function compiles with no errors, then indicate the output produced by the function?
(a)   int main() {
    Dessert a;
    Cookies b(2, 300);
    IceCream c("vanilla", 2, 350);
    return 0;
}

Does the program compile with no errors? Answer by Yes or No:  
If it does compile with no errors, what is the output produced by the execution?

(b)   int main() {
    Dessert* pc;
    IceCream* pi;
    pc = new Cookies(2, 250);
    pi = new IceCream("mint", 1, 250);
    delete pc;
    delete pi;
    return 0;
}

Does the program compile with no errors? Answer by Yes or No:  
If it does compile with no errors, what is the output produced by the execution?
(c) int main() {
    Sundae* ps;
    IceCream* pi;
    pi = new IceCream("mint", 1, 250);
    ps = pi;
    delete ps;

    return 0;
}

Does the program compile with no errors? Answer by Yes or No:  

If it does compile with no errors, what is the output produced by the execution?

(d) int main() {
    Dessert* ps = new IceCream("vanilla", 2, 350);
    cout << ps->cost() << endl;
    return 0;
}

Does the program compile with no errors? Answer by Yes or No:  

If it does compile with no errors, what is the output produced by the execution?

Amy (yup, the very good C++ programmer) wrote the class ListNode shown below. It represents a node in a linked list. It is similar to the one described in class and that you implemented in the labs.

```cpp
class ListNode {
    public:
        int id;
        ListNode* next;
};
```

Consider two linked lists. The first is pointed to by listA and the second is pointed to by listB.

```cpp
ListNode* listA;
ListNode* listB;
```

Normally the two lists contain separate nodes. However, it is possible due to some error for the two lists to share their nodes, as shown in the small example below.

```
listA
1 -> 2 -> 7 -> 9
|
8 -> 5
```

listB

Amy would like to write a function called `getCommon` that returns a pointer to the first node common to the two lists (node labeled C in the figure above). Otherwise, if there are no common nodes, the function returns NULL.

Amy decides to write this function with the help of a single additional data member in ListNode. This additional data member must be initialized by the constructor of ListNode and then used by the `getCommon` function. Amy would like to make this function run in O(n) as opposed to O(n^2).

(a) (1 mark). Declare the additional member to be added to ListNode here:

Answer:
(b) (1 mark). Show how this member should be initialized in the constructor of ListNode:

Answer:

(c) (3 marks). Write the body of the getCommon function below.

```c
ListNode* getCommon (ListNode* headA, ListNode* headB) {

// This is how getCommon is called
... = getCommon(listA, listB);
```
**Question 14. (5 marks). Binary Trees. (Caution: Difficult).**

Given a binary tree, you need to write a function to check if that tree is a mirror of itself. A tree is a mirror of itself if it is symmetric around its root.

For example the binary tree on the left below is a mirror. In contrast, the binary tree on the right below is not a mirror.

```
1
/ \
2 2
/ \
4 7 7 4
```

```
1
/ \ 2 2
/ \ 5 \ 5
```

Given the class definition of a TreeNode shown below, write a recursive function `isSymmetric` that returns true if the tree is a mirror, otherwise, it returns false.

```cpp
class TreeNode {
    public:
        int val;
        TreeNode *left;
        TreeNode *right;
};
TreeNode* root;
// This is how the function is called
isSymmetric(root->left, root->right);

bool isSymmetric(TreeNode *n1, TreeNode *n2) {
    if(n1 == NULL && n2 == NULL) return true; // If both nodes are null
    // Add your code here
}
```
Question 15. (4 marks). *Graphs.*

Consider the graph shown below.

(a) (2 marks). Give a *depth-first* traversal of the graph, starting at node 0. Assume that visiting a node means printing its number to the standard output.

Give the traversal here:  

(b) (2 marks). Give a *breadth-first* traversal of the graph, starting at node 0. Assume that visiting a node means printing its number to the standard output.

Give the traversal here:  
Question 16. (8 marks). *Complexity Analysis.*

Determine the *worst-case* time complexity (*expressed in big-O notation*) for each of the program segments below as a function of the size of the input n. Show the details of your analysis and clearly indicate your final result.

(a) (1 mark) The size of the input is \( n \).

```java
for (int i=0; i < n; ++i) {
    for (int j=0; j < 100000000 ; ++j) {
        // Some code with \( O(1) \)
    }
}
```

\( T(n) = \) 

(b) (2 marks) The size of the input is \( n \).

```java
for (int i=0; i < n; ++i) {
    for (int j=0; j < 3*n ; ++j) {
        // Some code with \( O(1) \)
    }
}
```

\( T(n) = \) 

(c) (2 marks). The size of the input is \( n \). Assume \( n \) is a power of two.

```java
for (int i=1; i <= n; i=i*2) {
    for (int k=0; k < n ; ++k) {
        // Some code with \( O(1) \)
    }
}
```

\( T(n) = \) 

(d) (3 marks). Assume for simplicity that $n$ is a power of two.

```c
int recursive(int n) {
    int x;
    int y;

    if (n <= 1) return 0;
    else {
        for (int i=0; i < n ; ++i) // Some code with O(1)
            recursive (n/2);
        recursive (n/2);
        return (1);
    }
}
```

Write the recurrence equation for $T(n)$.

Solve the recurrence equation (by expansion) to obtain an expression of $T(n)$ in terms of $n$.

Express $T(n)$ using the big-$O$ notation.

$T(n) = \boxed{\text{expression}}$
Question 17. (4 marks). Memory Management.

Consider the code below in a file called main.cpp. The line numbers shown are not part of the code and are there to facilitate your answer. The writer of this code suspects that it contains errors in how it uses memory, and runs valgrind on it. The output produced by valgrind is shown after the code and indicates several problems.

```
#include <iostream>
using namespace std;

class Mystery {
private:
    int *v;
    int size;
public:
    Mystery(int _size);
    ~Mystery();
    int& operator[] (int index);
    int getsize () const;
    void print () const;
};

Mystery::Mystery (int _size) {
    size = _size;
    v = new int[size];
}

Mystery::~Mystery () {
    delete v;
}

int& Mystery::operator[] (int index) {
    return (v[index]);
}

void Mystery::print () const {
    for (int i = 0; i <= getsize(); i++)
        cout << v[i] << " ";
    cout << endl;
}

int Mystery::getsize () const {
    return size;
}

int main () {
    Mystery a(5);
    for (int i=0; i < 5; i++) a[i] = i;
    a.print();
    return 0;
}
```
The valgrind output is as follows:

==10620== Memcheck, a memory error detector
==10620== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==10620== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==10620== Command: main.exe
==10620== Invalid read of size 4
==10620==  at 0x400A57: Mystery::print() const (in /u/user/amy/ece244/main.exe)
==10620==  by 0x400AEC: main (in /u/user/amy/ece244/main.exe)
==10620==  Address 0x4c45054 is 0 bytes after a block of size 20 alloc'd
==10620==  at 0x4A078F0: operator new[]{(unsigned long) (vg_replace_malloc.c:421)
==10620==  by 0x4009CB: Mystery::Mystery(int) (in /u/user/amy/ece244/main.exe)
==10620==  by 0x400AB4: main (in /u/user/amy/ece244/main.exe)
==10620==
==10620== Mismatched free() / delete / delete []
==10620==  at 0x4A06738: operator delete(void*) (vg_replace_malloc.c:574)
==10620==  by 0x4009F4: Mystery::~Mystery() (in /u/user/amy/ece244/main.exe)
==10620==  by 0x400AFD: main (in /u/user/amy/ece244/main.exe)
==10620==  Address 0x4c45040 is 0 bytes inside a block of size 20 alloc'd
==10620==  at 0x4A078F0: operator new[]{(unsigned long) (vg_replace_malloc.c:421)
==10620==  by 0x4009CB: Mystery::Mystery(int) (in /u/user/amy/ece244/main.exe)
==10620==  by 0x400AB4: main (in /u/user/amy/ece244/main.exe)
==10620==
==10620== All heap blocks were freed -- no leaks are possible
==10620== For counts of detected and suppressed errors, rerun with: -v
==10620== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 4 from 4)

In the table on the following page, indicate the changes required to the code to fix the bugs. A change must be one of: Remove, Replace with or Insert Before. **No changes should be made to the main() function; all fixes must be made with changes to the other functions.** Please list your fixes in increasing line order. You may or may not need all the rows in the table.
<table>
<thead>
<tr>
<th>Line Number</th>
<th>Remove / Replace with / Insert Before (choose one)</th>
<th>Code to Insert or Replace with</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In C++, an iterator is a class whose objects facilitate traversals of collections (i.e., data structures). Consider the following definition of a class for a linked list, similar to the one you used in your lab assignment 4 to maintain your list of asteroids, simplified for the purposes of the question.

class ListNode {
    public:
        int value;
        ListNode* next;
        // Other members that are not relevant to the question
    }

class List {
    public:
        ListNode* head; // Head of the list
        ListNode* begin(); // returns a pointer to the first term
        // on the list, NULL if list is empty
        ListNode* end(); // returns a pointer to the node
        // past the end of the list;
        // assume NULL
        // Other member that are not relevant to the question
    }
    List mylist;

In the main function, we wish to write the following code:

for (ListIterator it = mylist.begin(); it != mylist.end(); it++) {
    cout << *it << endl;
}

This allows the main function to traverse the list with no knowledge of how the list is implemented.

However, in order for the above code to work, a class called ListIterator is defined.

class ListIterator {
    public:
        // One data member is defined here
        // One constructor is defined here
        // Some methods are defined here
    };
(a) (1 marks). The `ListIterator` class must define one data member. Show the declaration of this data member here.

**Answer:**

(b) (2 marks). The `ListIterator` class must define one constructor. Show the implementation of this constructor below. Make sure to include both the header and the body of the constructor.

```java
{
Write method header here
Write method body here
}
```

(c) (8 marks). The `ListIterator` class must define four overloaded operators. Show the implementation of these overloaded operators below. For each, make sure to include both the header and the body of the operator. Assume them all to be members of the class `ListIterator`.

```java
{
Write method header here
Write method body here
}
```