

THE AUSTRALIAN NATIONAL UNIVERSITY
Final Examinations(Semester 2) 2006

COMP1110/COMP1510
(Introduction to Software Systems/Introduction to Software Engineering)
Final Exam

Writing Period: 2 hours duration
Study Period: 15 minutes duration
Permitted Materials: One A4 page with notes on both sides.

Name (Family name, First name)

Student Number

Subject code

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY.

- This paper will be marked out of 100 and consists of 5 questions. Questions are of unequal value. The value of each question is shown in square brackets. Questions that are partitioned into parts show the number of marks given to each part within square brackets. There are 2 versions of Question 5: one for comp1110 students only, and the other for comp1510 students only. Students should attempt the question based on the subject they are enrolled in only. Students should attempt all other questions.
- Answer all questions using either a black or blue pen. Use the space provided. Marks may be lost for giving information that is irrelevant. There is additional space at the end of this booklet in case the space provided is insufficient (Clearly indicate, within the question concerned, that you have used this extra space at the end of the booklet.).
- Students are asked to check that this examination paper contains all 18 pages. No pages are to be torn from this examination paper.
- This examination paper is **CONFIDENTIAL** and is not to be taken from the examination room.

Official use only:

1	2	3	4	5	
---	---	---	---	---	--

Question 1. [10 Marks]

Given the following code:

```
import java.util.ArrayList;
public class Quiz {
    static String rec(int i) {
        return (i<=0 ? "X" : "(" + rec(i-1) + ")" + rec(i-2));
    }
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>();
        list.add("Hello");
        list.add("Bye");
        System.out.println("list : " + list);
        System.out.println(list.get(1));
        System.out.println(list.get(0));
        list.add(1, "End");
        System.out.println("list : " + list);
        list.remove("Hello");
        System.out.println("list : " + list);
        list.set(0, "Done");
        System.out.println(list.get(1));
        System.out.println(list.size());
        System.out.println(rec(0));
        System.out.println(rec(1));
        System.out.println(rec(4));
    }
}
```

Write exactly what the above program will output if it was run.

Question 2. [10 Marks]

Given the below 15 True/False questions, circle either **T** or **F** (but not both). Each question that is correctly answered gains you 1 mark, each question answered incorrectly loses you 1 mark, a question left unanswered neither loses nor gains marks. The final mark for this question is calculated by bounding the sum of marks between 0 and 10.

For example, if you answered all questions correctly you would gain 10 (not 15) for this question.

If you answer 8 correctly, 2 incorrectly and leave the remaining 5 unanswered you would gain 6/10 for this question.

If you wish to change your answer then cross off both **T** and **F** and clearly state your answer in words. (e.g. "I would like to leave this question unanswered." or "My answer is True.")

The following True/False questions relate to ArrayList:

T F : The first element in an ArrayList has index 1.

T F : ArrayLists only grow.

T F : 'new ArrayList(10)' will create an ArrayList of size 0.

T F : Java implements the ArrayList class using linked lists.

The following True/False questions relate to software engineering:

T F : The waterfall model involves moving from one stage to the next in sequence.

T F : The software life cycle does not include delivery and maintenance.

T F : Frederic Brooks thought conceptual integrity was important in system design.

The following True/False questions relate to java:

T F : By convention, variable names should start with an uppercase letter.

T F : By convention, class names should start with a lowercase letter.

T F : The assignment operator (=) is used to evaluate equality of two expressions.

The following True/False questions relate to loops in java:

T F : The loop, "for (Integer i = 1; i < 10; i++) {...}", would execute the block of code 10 times.

T F : The loop, "for (Integer i = 0; i <= 10; i++) {...}", would execute the block of code 10 times.

T F : The 'for each' statement is useful if you wish to delete elements of a collection as you iterate over the collection.

T F : 'while' statements execute the block of code zero or more times.

T F : 'for' statements always execute the block of code at least once.

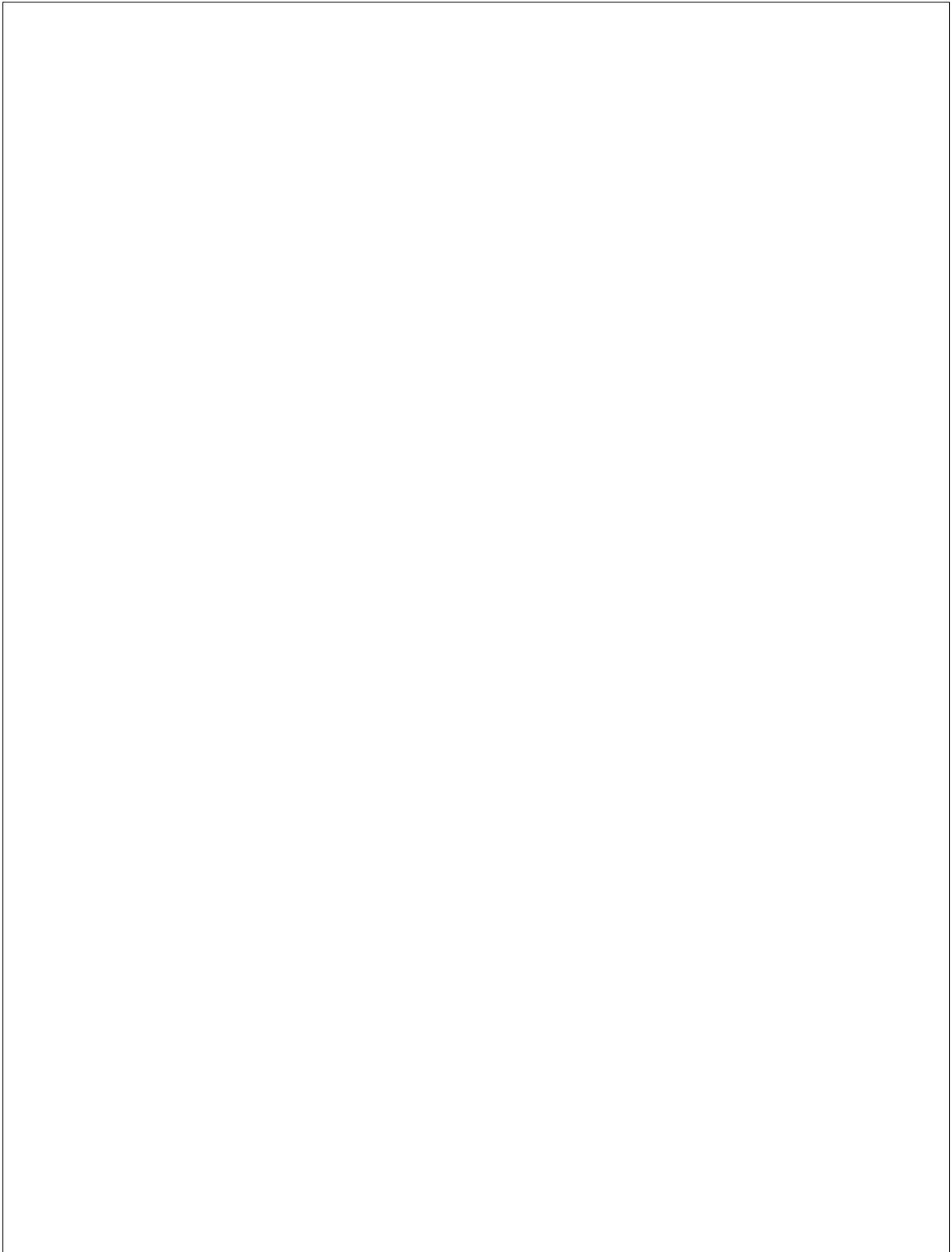
Question 3. [40 Marks] Suppose you are given the task of writing a program that helps a radio station manage and analyse their daily list of songs and advertisements. Each day a list of playable items is constructed. These playable items are either songs or advertisements. Songs have : a title, the name of the band, and the number of seconds it takes to play the song(this includes any time it takes to announce the song). Advertisements have : the number of seconds it takes to play the advertisement, the company name the advertisement comes from, and the amount of money the radio station will earn from playing the advertisement. Part of a day's play list may look like:

- Song - title : *Vertigo* band : *U2* time : *210 sec*
- Song - title : *Bad* band : *Michael Jackson* time : *230 sec*
- Advertisement - company : *AGL* time : *30 sec* earn: *\$320.00*
- Advertisement - company : *Holden* time : *20 sec* earn: *\$210.00*
- Song - title : *Little Lamb* band : *Mary* time : *210 sec*
- *etc...*

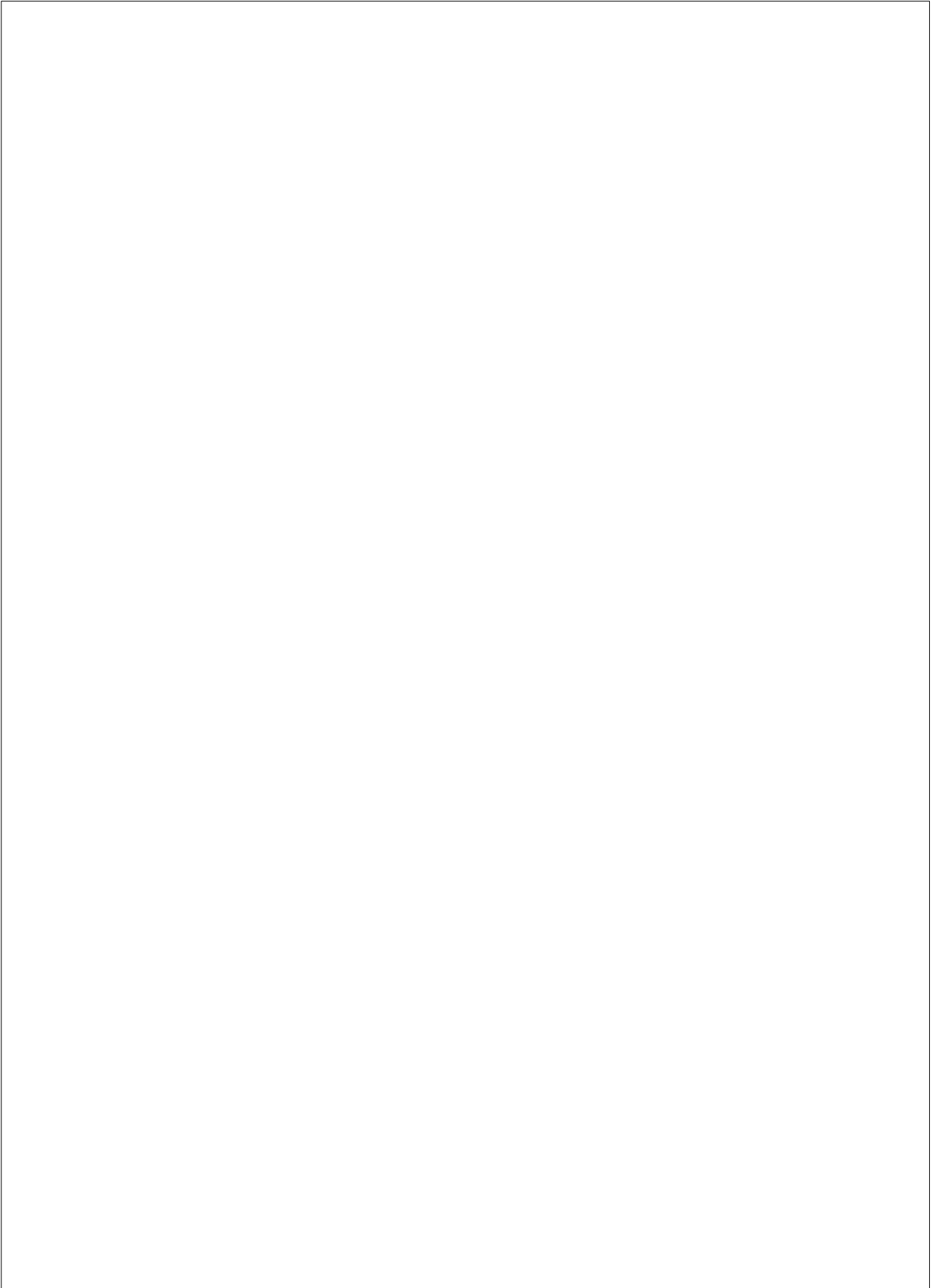
The radio station manager wishes to be able to analyse a day's play list to help work out the content for the day. Your program must include 3 methods: 'totalTime' which calculates the total time in seconds to play all of the day's content, 'totalEarnings' which is the sum of the amount of money that will be made from the advertisements on that day, and 'tooManyAds' which returns true if and only if there are too many advertisements in any one hour. (The radio station is only permitted to have at most 12mins of advertisements in any one hour.) You are not required to parse the play list from a file or provide an interface for designing the play list, also there is no need to implement constructors.

[5 Marks] i) What question(s) would you ask your client to help refine this specification? State any reasonable assumptions you make to answer these question(s). Use these assumptions for the following parts of this question.

[10 Marks] ii) Draw the UML Class diagram for your design. Include fields and methods in the class diagram.



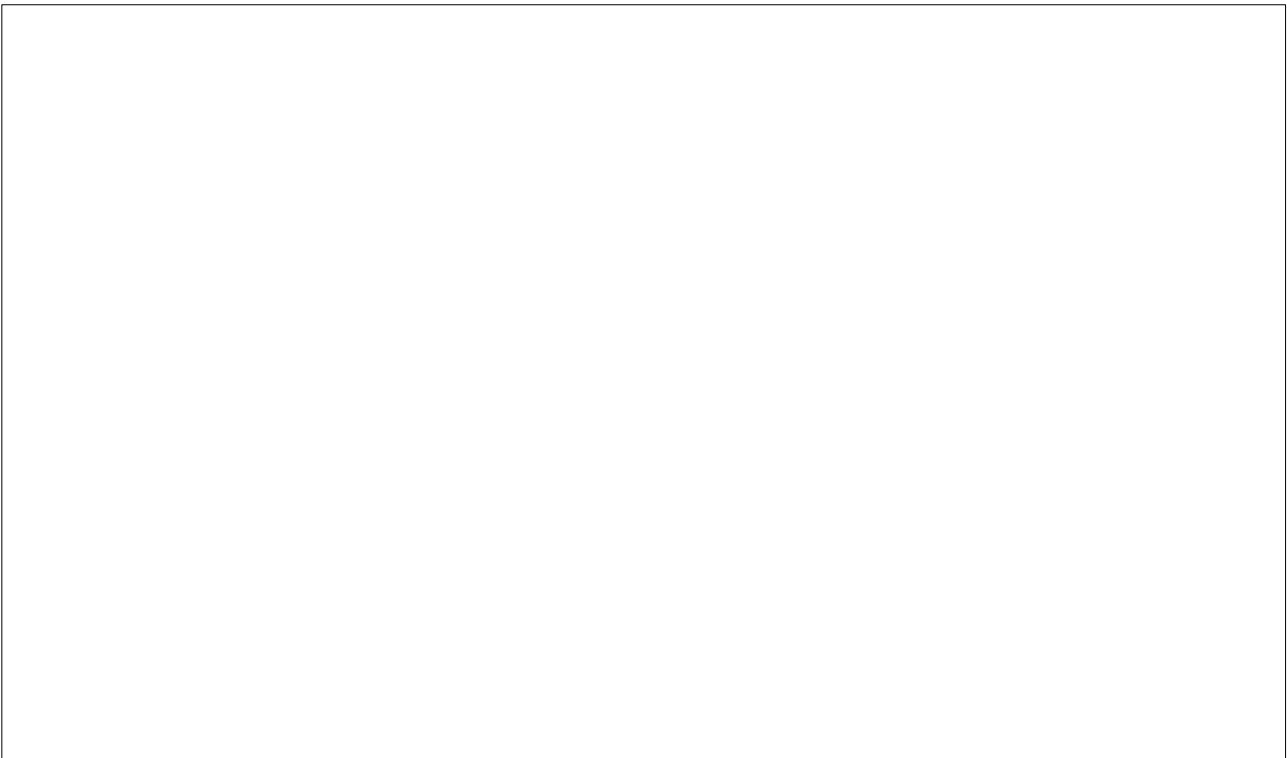
[10 Marks] iii) Implement the classes stated in your design. Your implementation should include the 3 methods stated in the question's description along with any other methods you need to implement these 3 methods.



[5 Marks] iv) State what test cases you would use to help validate the "tooManyAds" method.



[10 Marks] v) Use big O notation to analyse the worst case complexity of the "totalEarnings" and "tooManyAds" methods.



Question 4. [30 Marks]

[12 marks] i) Imagine we're using linked lists to represent sets, in such a way that there might be several occurrences of the same element in the list. In order to print out a nice representation of the set, and for efficiency reasons, we'd like to go through the list and remove all but one instance of each element in the list.

Given the following implementation of an immutable linked list of String:

File IList.java:

```
public interface IList {
    public String getFirst();
    public IList getRest();
    public IList addFirst(String s);
    public Boolean has(String s);
}
```

File EmptyList.java:

```
import java.util.NoSuchElementException;

public class EmptyList implements IList
{
    public String getFirst() {
        throw new NoSuchElementException();
    }
    public IList getRest() {
        throw new NoSuchElementException();
    }
    public IList addFirst(String s) {
        return new NEList(s, this);
    }
    public Boolean has(String s) {
        return false;
    }
}
```

File NEList.java:

```
public class NEList implements IList {
    private String data;
    private IList rest;

    public NEList(String head, IList tail) {
        data = head;
        rest = tail;
    }
    public String getFirst() {
        return data;
    }
    public IList getRest() {
        return rest;
    }
    public IList addFirst(String s) {
        return new NEList(s, this);
    }
    public Boolean has(String s) {
        if (data.equals(s)) {
            return true;
        } else {
            return rest.has(s);
        }
    }
}
```

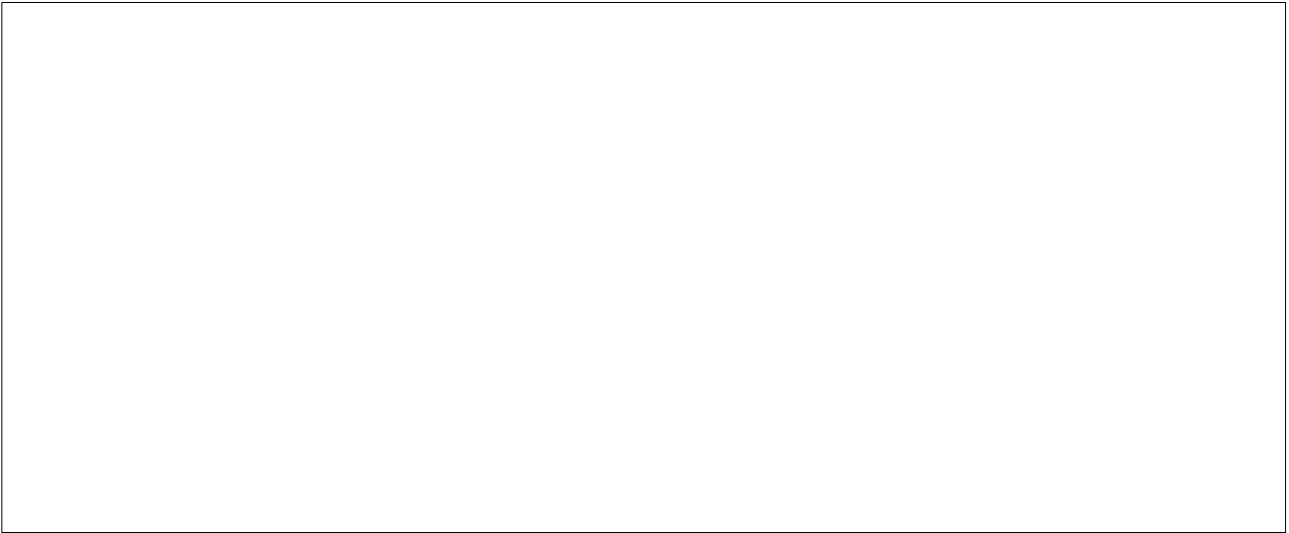
Write a method:

```
public IList removeDuplicates()
```

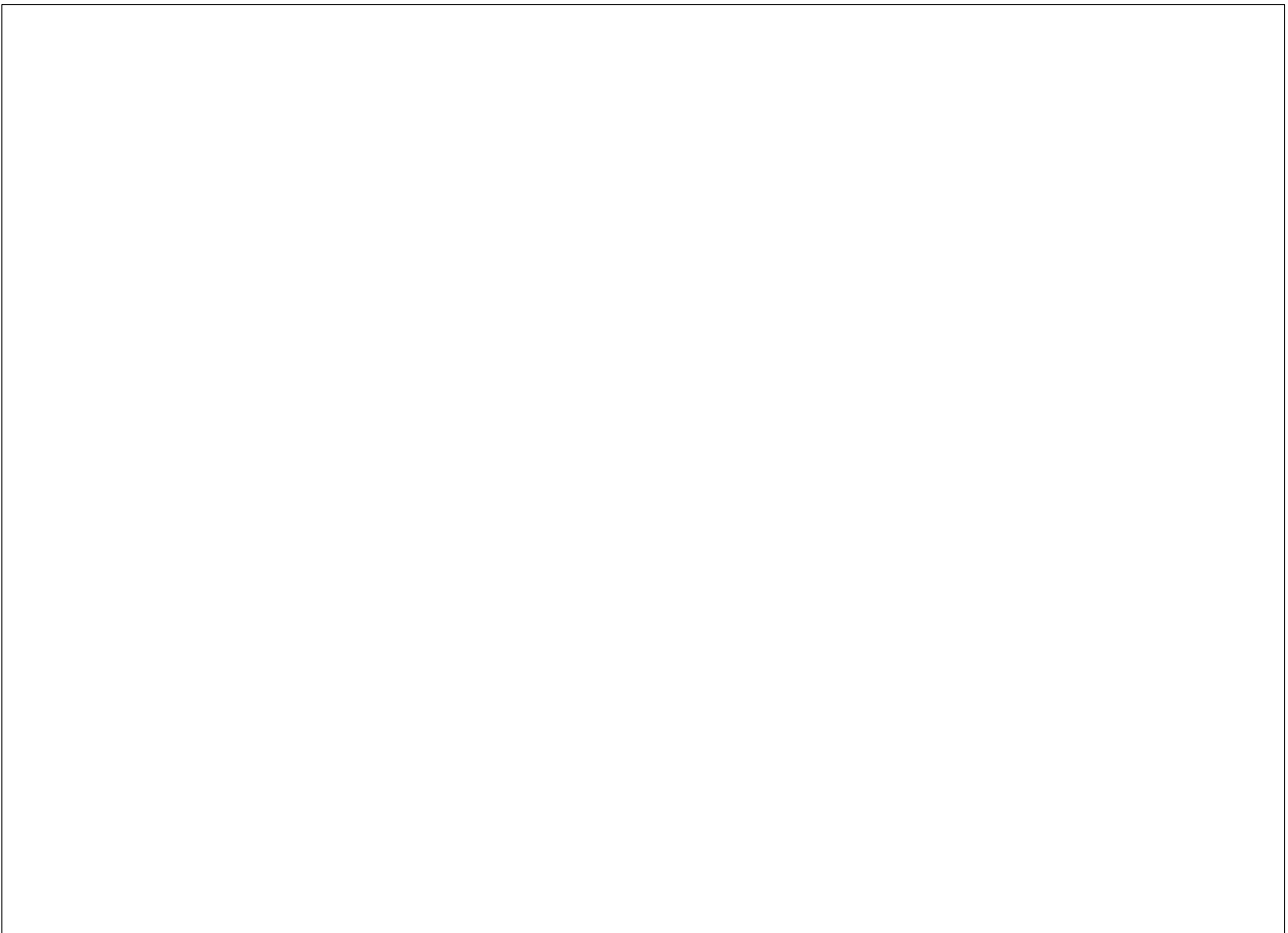
that removes all duplicates from a list, so that no element appears twice in the result, but every element that was there originally is still there. Clearly state additions made to each of the files. (Hint: First write a method, `IList removeAll(String s)`, that removes all occurrences of a given string from a list.)

Questions (ii.a) - (ii.e) relate to binary search trees.

[2 mark] (ii.a) What are the conditions for a binary tree to be a binary search tree?



[3 marks] (ii.b) To check whether a binary tree satisfies the conditions for a binary search tree, why is it **not** enough to check each node against its two immediate children? Draw a diagram of a binary tree that looks OK at each node, but is **not** a valid binary search tree.



Given the following minimal implementation of binary search trees (of integers).

File BSTree.java:

```
public interface BSTree {
    public BSTree insert(Integer item);
}
```

File EmptyTree.java:

```
public class EmptyTree implements BSTree {
    public BSTree insert(Integer item) {
        return new NETree(item, new EmptyTree(), new EmptyTree());
    }
}
```

File NETree.java:

```
public class NETree implements BSTree {
    private Integer root;
    private BSTree left;
    private BSTree right;

    public NETree(Integer root, BSTree left, BSTree right) {
        this.root = root;
        this.left = left;
        this.right = right;
    }
    public BSTree insert(Integer item) {
        if (item == root) {
            return this;
        } else if (item < root) {
            return new NETree(root, left.insert(item), right);
        } else {
            return new NETree(root, left, right.insert(item));
        }
    }
}
```

[3 marks] (ii.c) Add a method:

```
public Integer height()
```

that returns the height of the tree. State clearly what changes you would make to each of the classes above.

[4 marks] (ii.d) Add a method

```
public Boolean has(Integer target)
```

that returns true if the target value is in the tree and false otherwise. State clearly what changes you would make to each of the classes above.

[2 marks] (ii.e) Explain why the complexity of *has()* in a binary search tree is $O(n)$ in the worst case. Under what circumstances is it $O(\log_2(n))$?

Questions (iii.a) - (iii.c) relate to hash tables.

[1 mark] (iii.a) What is a **collision** in a hash table?

[1 mark] (iii.b) Suppose a hash function always returns an even number. What is wrong with this?

[2 marks] (iii.c) Explain what **linear probing** is in a hash table implementation.

Question 5. (COMP1110 Only) [10 Marks]

[4 marks] (i) Give at least two advantages of adding JML contracts to Java code.

[2 mark] (ii) What is a **pure method**?

[4 marks] (iii) Look at the following JML contract for a square root method.

```
/*@ requires x >= 0.0;  
   ensures Math.abs(\result * \result - x) < 1.0e-6; */
```

What responsibilities does this contract impose on the caller of the method? What responsibilities does it impose on the method implementation?

Question 5. (COMP1510 Only). [10 Marks]

Within applied ethics what is Consequentialism? Provide an illustration of an ethical problem within software engineering. Provide an argument a utilitarian may make regarding this ethical problem. How would this differ from some deontological ethic.

Additional space

Additional space