UNIVERSITY OF NEW ENGLAND

NAME:_____

STUDENT NUMBER:_____

UNIT NAME: COMP132

PAPER TITLE: Computer Science II

PAPER NUMBER: First and Only

DATE: **Wednesday 23 June 2010** **TIME:** **1:45 PM TO 4:00 PM**

TIME ALLOWED: Two (2) hours and fifteen minutes

NUMBER OF PAGES IN PAPER: FIFTEEN (15)

NUMBER OF QUESTIONS ON PAPER: NINE (9)

NUMBER OF QUESTIONS TO BE ANSWERED: NINE (9)

STATIONERY PER CANDIDATE:					
0	6 LEAF A4 BOOKS	**1**	ROUGH WORK BOOK	**0**	GENERAL PURPOSE ANSWER SHEET
1	12 LEAF A4 BOOKS	**0**	GRAPH PAPER SHEETS	**0**	SEE OTHER 'AIDS REQUIRED' BELOW

OTHER AIDS REQUIRED: **NIL**

POCKET CALCULATORS PERMITTED: **NO**

TEXTBOOKS OR NOTES PERMITTED: **NIL**

INSTRUCTIONS FOR CANDIDATES:

- Candidates MAY NOT start writing until instructed to do so by the supervisor
- Please pay attention to the announcements and read all instructions carefully before commencing the paper
- Candidates MUST write their name and student number on the top of this page
- Answer all questions
- Questions are NOT of equal value
- This examination question paper **MUST BE HANDED IN** with worked scripts. Failure to do so may result in the cancellation of all marks for this examination

REMEMBER TO WRITE YOUR NAME AND STUDENT NUMBER AT THE TOP OF THIS PAGE

Question 1

Multiple Choice: Choose the one alternative that best completes the statement or answers the question.

(a) *[2 marks]*

Given the following code:

```
public static int TestIt(int x, int y)
        if ( x < y)
        {
        return -5;
        }
        else
        {
        return (TestIt(x - y, y + 5) + 6);
        }
}
```

What value is returned for the following method call:

TestIt(10, 20)

(i) 10

(ii) -5

(iii) 1

(iv) 6

Question 1 (b) is on page 3

(b) *[2 marks]*

The depth of recursion is

(i) The value returned from the last recursive call

(ii) The number of times that a method calls itself

(iii) The value that will terminate the recursive calls

(iv) There is no such term

(c) *[2 marks]*

If an algorithm makes two separate, unnested passes over an input of size n, the performance of the algorithm will be in the class

(i) $O(n)$

(ii) $O(2)$

(iii) $O(2^{n^2})$

(iv) $O(n^2)$

(d) *[3 marks]*

The selection sort algorithm works by

(i) repeatedly comparing adjacent items and swapping them so smaller values come before larger values

(ii) repeatedly taking the first value in the unsorted portion of the array and placing it at its proper place in the part of the array that is already sorted

(iii) repeatedly locating the smallest value in the unsorted portion of the array and moving it toward the lower end of the array

(iv) partitioning the unsorted portion of the array into two sublists and a pivot and recursively sorting the two sublists

Question 1 (e) is on page 4

(e) *[2 marks]*

Let `Point<T>` be a generic type. We want to write a method that takes as parameter Point objects whose type parameter is the Number class, or any superclass of Number. We can do this by writing

(i) Point<? sub Number>

(ii) Point<? super Number>

(iii) Point<? extends Number>

(iv) Point<Number>

(f) *[2 marks]*

Which of the following statements are true?

(i) You cannot create arrays whose elements are instances of a generic type

(ii) You cannot instantiate an object of a generic type

(iii) You can declare references to arrays whose elements are of a generic type

(iv) All of the above

(g) *[2 marks]*

A collection suitable for use in an application with multiple threads accessing a list is a(n)

(i) ArrayList

(ii) HashSet

(iii) Map

(iv) Vector

Question 1 (h) is on page 5

(h) *[2 marks]*

Which of the following is true?

(i) A TreeSet created with the no-arg constructor expects its elements to implement the Comparable interface

(ii) The load factor of Tree Set should never exceed 50% to ensure maximum performance

(iii) The load factor of Tree Set should never exceed 75% to ensure maximum performance

(iv) A TreeSet created with the no-arg constructor expects its elements to implement the Comparator interface

(i) *[2 marks]*

Assume an array-based list implemented by a class that uses the fields

```
String [ ] list;
int nElements;
```

to represent the array of elements, and the number of elements currently stored. Assuming there is space in the internal array, the code for adding a new item str to the end of the list is

(i) `list.add(str);`

(ii) `nElements ++;  list[nElements] = str;`

(iii) `list[nElements] = str;`

(iv) `list[nElements] = str;  nElements++;`

Question 1 (j) is on page 6

(j) *[2 marks]*

If a new element is added to an ArrayList whose internal array if already full,

(i) the new element is not added, and null is returned

(ii) the add method throws an exception

(iii) A new, bigger internal array is created, the elements are moved to the new array, the old internal array is deleted, and then the new element is added

(iv) the new element is not added, and -1 is returned

(k) *[2 marks]*

When using recursion on linked lists

(i) the recursive method should be made private, and should be called by a public non-recursive method

(ii) the recursive method should be one of the methods specified in the List interface

(iii) the recursive method should not call itself outside of its base case

(iv) the linked list class is subclassed, and then the recursive method overrides a method of the same name in the list class

(l) *[2 marks]*

A **circularly linked** list makes it easy to

(i) jump from the last node to the first

(ii) move from any node to its predecessor

(iii) jump from node to node

(iv) move from any node to its successor

Question 1 (m) is on page 7

(m) *[2 marks]*

A stack based on a linked list is based on the following code

```
class Node{
        String element;
        Node next;
        Node(String el, Node n)
        {
        element = el;
        next = n;
        }
}
Node top = null;
```

The code for testing whether the stack is empty is

(i)
```
return top = null;
```

(ii)
```
if (top == null)
        return true;
    else throw new RunTimeException();
```

(iii)
```
return top == null;
```

(iv)
```
if (top == 0)
        return true;
    else
        return false;
```

(n) *[2 marks]*

A stream of cars going through a toll booth is an example of a

(i) stack

(ii) priority queue

(iii) queue

(iv) hash set

(o) *[2 marks]*

If "14t8" is entered for input in the following code,

```
while (input != null)
{
        try
        {
        totalIncome += Double.parsedouble(input);
        months++;
        }
        catch(NumberFormatException e)
        {
        System.out.println("Non-numeric data encountered in the file: "
        + e.getMessage());
        }
        input = inputFile.readLine();
}
```

what does the program do?

(i) input will cause a NumberFormatError, the catch clause will be executed, then the program will resume with the statement following the while statement

(ii) input will be converted to a double and added to totalIncome, months will be incremented by 1, and the while statement will be repeated until a null is entered

(iii) input will cause a NumberFormatError, the catch clause will be executed, then the program will continue by asking for the next input value

(iv) input will cause a NumberFormatError, the catch clause will be executed, then the terminate

(p) *[2 marks]*

When a new item is added to an **AVL tree**

(i) each node has at most one predecessor and at most one successor

(ii) each node has at most one predecessor and exactly two successors

(iii) each node has at most one predecessor and at most two successors

(iv) each node has at least one predecessor and at most two successors

Question 2

Exceptions

(a) *[4 marks]*

Describe what an exception is.

(b) *[4 marks]*

How do you retrieve an error message from an exception?

(c) *[4 marks]*

What are the differences between a checked and an unchecked exception?

Question 3 is on page 10

Question 3

Generics *[3 marks]*

Consider the class

```
class Value <T extends Number>
{
    private T v;
    public Value(T v1)
    {
    v = v1;
    }
    public void output()
    {
    System.out.println(v);
    }
}
```

What will the following statement do?

```
    Value<Double> nV1 = new Value<Double>(34.5);
```

Question 4 is on page 11

Question 4

Recursion *[3 marks]*

Why are recursive algorithms usually less efficient than iterative algorithms?

Question 5

Collections

(a) *[3 marks]*

Why is it common practice to use an interface variable to reference a collection?

(b) *[3 marks]*

A linked list has some advantages and disadvantages over an array.
What are the advantages and disadvantages?

(c) *[2 marks]*

The following code creates an ArrayList to hold String objects:

```
List<String> nameList = new ArrayList<String>();
```

Given the following code:

```
ListIterator<String> it = nameList.listIterator();
```

Why do you **NOT** need to instantiate the list iterator yourself?

(d) *[2 marks]*

What is a hash code?

(e) *[2 marks]*

In hashing, what is a collision?

Question 6

Array-Based Lists

(a) *[3 marks]*

Assume an array-based list implemented by a class that uses the fields

```
String [ ] list;
int nElements;
```

to represent the array of elements, and the number of elements currently stored.

Write the code for a method *int size()* that returns the current size of the list.

(b) *[6 marks]*

Briefly describe the operations specified by the `Iterator` interface.

(c) *[4 marks]*

What is the difference between the Iterator and Iterable interfaces?

Question 7

Linked Lists *[6 marks]*

A linked list class uses a Node class with successor reference *next* and field *element* to store values. It uses a reference *first* to point to the first node of the list.

Correct the following code (rewrite it) so that it prints all elements stored in the list

```
Node p  = first;
while (p != null)
System.out.println(p.next);
```

Question 8

Stacks and Queues *[6 marks]*

Consider a class that uses the following variables to implement an array-based stack:

```
String [ ] s = new String[100];
int top = -1;   //Note  top == -1 indicates stack is empty
```

Write the code for a method that implements a push operation. The method header for this operation is:

void push(String x)

(Note if the stack has no more room it should show the appropriate exception)

Question 9 is on page 14

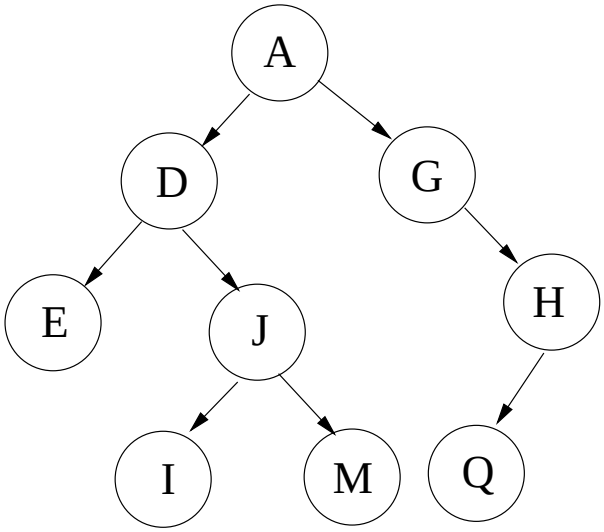
Question 9

Trees

(a) *[6 marks]*

The operation of "tree traversal" can be simply stated as the movement through a binary tree in such a way that each node of the tree is visited exactly once.

Given the following binary tree:



Write the order of the nodes visited using the following traversal methods:

a) preorder

b) postorder

c) inorder

Question 9 (b) is on page 15

(b) *[6 marks]*

Assuming a Node class

```
class Node
{
    int element;
    Node left, right;
    Node(int el, Node left, Node right)
    {
    element = el;
    this.left = left;
    this.right = right;
    }
}
```

Write a method, with the following method header

int size(Node tree)

that returns the number of nodes in the binary tree whose root is *tree*

Please remember - This examination question paper MUST BE HANDED IN. Failure to do so may result in the cancellation of all marks for this examination. Writing your name and number on the front will help us confirm that your paper has been returned

.