IVE Information Technology

Information & Communications Technology

Programme Board

HIGHER DIPLOMA IN

SOFTWARE ENGINEERING

(IT114105)

MODULE TITLE:

DATA STRUCTURES & ALGORITHMS: CONCEPTS AND IMPLEMENTATION

MODULE CODE: **ITP4510**

SEMESTER TWO MAIN EXAMINATION

6 MAY, 2016
1:30 PM TO 3:30 PM (2 hours)

Instructions:

(a) This paper has a total of TWELVE pages including the covering page.

(b) This paper contains TWO Sections.

(c) Section A is WORTH 40 marks and Section B is WORTH 60 marks.

(d) Section A contains THREE questions. Answer ALL questions in Section A.

(e) Section B contains THREE questions. Answer ALL questions in Section B.

Note: The result of this assessment will not be counted if you do not meet the minimum attendance requirement (if any) governed by the general academic regulations of your programme/course unless approval of the campus principal has been granted.

This paper contains TWO sections.

Section A (40 marks)

This section contains 3 questions.

Answer ALL questions.

A1 Consider the following Java program:

```
abstract class Eatable {

    String name;
    String unit;
    double price;

    public Eatable(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public abstract double totalSugar();
    public abstract double getPrice();
}

class Drinks extends Eatable {

    double sugar;  //sugar per 100ml
    double volume;

    /*** CONSTRUCTOR TO BE COMPLETED IN A1 part (a) ***/

    public String toString() {
        return name + " : " + volume + unit + ", price=$" +
                    getPrice() + ", sugar=" + totalSugar() + "g";
    }
}

public class FoodAndDrinks {

    public static void main(String[] args) {
        Drinks d1 = new Drinks("coke", 5.5, 11.0, 330);
        System.out.println(d1);
        Drinks d2 = new Drinks("orange juice", 9.0, 8.5, 430);
        System.out.println(d2);
        Drinks d3 = new Drinks("coffee", -999.9, 7.0, 250);
        System.out.println(d3);
    }
}
```

*** Question A1 continues in next page ***

***** Question A1 continues from previous page *****

The above program generates the following screen output:

```
coke : 330.0ml, price=$5.5, sugar=36.3g
orange juice : 430.0ml, price=$9.0, sugar=36.55g
coffee : 250.0ml, price=$3.0, sugar=17.5g
```

Note:

- The *unit* of *Drinks* are always "*ml*".
- *Sugar* in *Drinks* class stores the sugar in grams per 100ml of the drink.

(a) Write the constructor of class *Drinks*. [7 marks]

(b) Write the missing methods *totalSugar()* and *getPrice()* in class *Drinks* to override the abstract methods in class *Eatable*. Apply the following rule while getting the price out from *Drinks* objects. [6 marks]

- The *price* of drinks must be more than or equal to $3.0.

(c) Will the following code run? Explain your answer. [2 marks]

```
Eatable e1 = new Eatable("Burger", -500.0);
```

A2 Consider the following Java program:

```
abstract class Animal {

    String name;
    double weight;

    public Animal(String name, double weight) {
        this.name = name;
        this.weight = weight;
    }
    public double getWeight() {
        return weight;
    }
    public void setWeight(double weight) {
        this.weight = weight;
    }

} // end of class Animal


/*** INTERFACE Live TO BE COMPLETED IN A2 part (a) ***/
// …


class Dog extends Animal implements Live {

    public Dog(String name, double weight) {
        super(name, weight);
    }
    public void Speak() { System.out.println(name + ": arf!");  }

    public void Eat() { System.out.println(name + ": slurp!");  }

    public void Speak(String content) { System.out.println(name +
                        ": " + content + " arf!"); }

} // end of class Dog


class Cat extends Animal implements Live {

    public Cat(String name, double weight) {
        super(name, weight);
    }
    public void Speak() { System.out.println(name + ": meow!"); }

    public void Eat() { System.out.println(name + ": glup..."); }

    public void Speak(String content) { System.out.println(name +
                        ": " + content + " meow!"); }

}  // end of class Cat
```

*** Question A2 continues in next page ***

***** Question A2 continues from previous page *****

```
public class Zoo {

    public static void main(String[] args) {

        Live a[] = new Live[2];
        a[0] = new Dog("Odie", 15);
        a[1] = new Cat("Garfield", 8);

        a[0].Speak();
        a[1].Speak("Today is a sunny day!");
        a[0].Speak("Let's go and chase the postman!");
        a[1].Speak("I rather stay home and eat.");
        a[1].Eat();
        a[0].Speak("Let's get out!!");
        a[1].Eat();
        a[0].Eat();
    }

}  // end of class Zoo
```

(a) Write the interface *Live*. [4 marks]

(b) Write down the output of the above program. [4 marks]

(c) The name of the *Animal* objects (*Garfield* and *Odie*) cannot be modified once the objects are created. Suggest a way to allow the modification after the objects have been created. [2 marks]

A3 Consider the following Java program implementing Singly Linked Lists:

```
class Node {
    private Object data;
    private Node next;
    public Node getNext() {
        return next;
    }
    public void setNext(Node next) {
        this.next = next;
    }
    public Node(Object data, Node next) {
        this.data = data;
        this.next = next;
    }
    public Object getData() {
        return data;
    }
}

class LinkedQueue {
    private Node head;
    private Node tail;
    private int size = 0;  // size of the queue

    public int getSize() {
        return size;
    }
    public boolean isEmpty() {
        return (size == 0);
    }
    public void enqueue(Object item) {
            /* to be completed */
    }
    public Object dequeue() throws QueueEmptyException {
        if (size == 0)
            throw new QueueEmptyException();
        else {
            Node n = head;
            head = head.getNext();
            size--;
            return n.getData();
        }
    }
    public String toString() {
        String s = "[";
        Node n = head;
        while (n != null) {
            s += n.getData().toString() + " ";
            n = n.getNext();
        }
        return s + "]";
    }
}
```

*** Question A3 continues in next page ***

***** Question A3 continues from previous page *****

(a) Write the *enqueue* method. [5 marks]

(b) Write down the output of the program below. [3 marks]

```
    public static void main(String[] args) {

            LinkedQueue q1 = new LinkedQueue();
            q1.enqueue("Zoe");
            q1.enqueue("Dan");
            q1.enqueue("Ida");
            System.out.println(q1);

            q1.enqueue("Leo");
            q1.dequeue();
            q1.dequeue();
            System.out.println(q1);

            q1.enqueue("Liz");
            q1.enqueue(q1.dequeue());
            System.out.println(q1);
    }
```

(c) Write down the output of the program below. [4 marks]

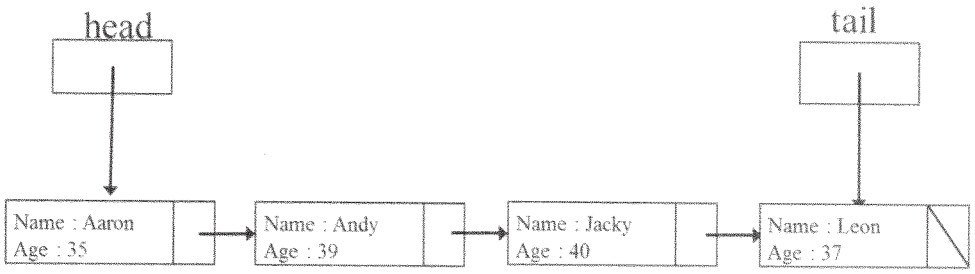
```
public static void main(String[] args) {
        try {
            LinkedQueue q2 = new LinkedQueue();
            q2.enqueue("Ariel");
            System.out.println(q2);
            q2.dequeue();
            System.out.println(q2);
            q2.dequeue();
            System.out.println(q2);

        }
        catch (QueueEmptyException e) {
            System.out.println("QueueEmptyException caught");
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException caught");
        }
        catch (Exception e) {
            System.out.println("Exception caught");
        }
        finally {
            System.out.println("Program ends");
        }
}
```

(d) Assuming the codes in *parts (b)* and *(c)* above can be compiled and run correctly. Is *QueueEmptyException* checked or unchecked exception? Explain your answer. [3 marks]

Section B (60 marks)

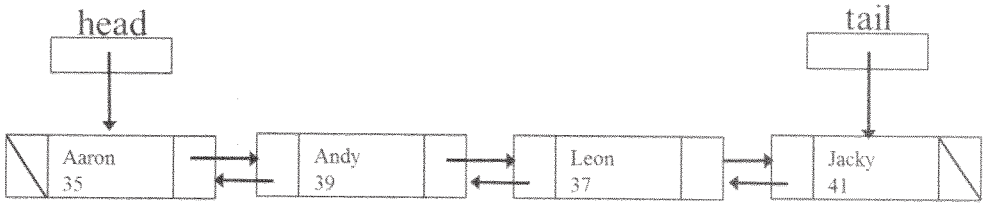
This section contains 3 questions. Each question carries 20 marks.
Answer ALL questions.

B1 (a) Given the following diagram to show a singly linked list with four Person nodes. The nodes were inserted according to ascending alphabetic order of the "Name" field.



Describe the steps to delete a Person whose name is "Jacky". [4 marks]

(b) Given the following illustration of doubly Linked List and implementation:




```
class DoublyNode {
  String name;  int age;
  DoublyNode next, previous;
  DoublyNode(String n, int a) { /*** TO BE COMPLETED in B1 part b(i) ***/ }
  DoublyNode(String n, int a, DoublyNode nextNode,
             DoublyNode previousNode)
   { /*** TO BE COMPLETED in B1 part b(i) ***/ }
  //other methods, such as addToHead(…), removedFromHead(), etc.
} // end of class DoublyNode

class DoublyLL {
  private DoublyNode head, tail;
  public DoublyLL( ){ head=tail=null; }
  public boolean isEmpty( ){ return head==null; }
  …
} // end of class DoublyLL
```

***** Question B1 continues in next page *****

***** Question B1 continues from previous page *****

(i) Complete the constructors of the DoublyNode class. [4 marks]

(ii) Given the following removeFromTail( ) method implemented in a singly linked list class. Re-write the **"else"** part to fit for the doubly linked list class.

[2 marks]

```
public Object removeFromTail()throws EmptyListException {
        ...
    if (head == tail)
       head = tail = null;
    else {
       ListNode current = head;
       while (current.next != tail)
          current = current.next;
       tail = current;
       current.next = null;
      }
      ...
}
```

(iii) Extra costs of double linked list over the single linked list is the space and time requirement. Explain what type of application worth spending the extra costs to use the double linked list. [3 marks]

(c) Given the following LinkedList class:

```
class LinkedList {
   private ListNode head, tail;
   public LinkedList() {…}
   public boolean isEmpty() {…}
          ...
   public void addToTail(Object obj){…}
   public Object removeFromTail( ) throws EmptyListException {…}
   public void addToHead(object obj) {…}
   public Object removeFromHead() throws EmptyListException {…}
}
```

Complete the following missing segments (A) to (E) in your answer book by using the above LinkedList class together with object-oriented inheritance skill. [7 marks]

```
class ListStack  … missing segment (A)….  {
   public ListStack( ) { super( ); }
   public boolean empty( ) { … missing segment (B)…}
   public void push(Object item) { … missing segment (C)… }
   public Object pop() { … missing segment (D)… }
   public Object peek()//return top element without removing it.
      { … missing segment (E)… }
} // end of class ListStack
```

B2 Assume that a Binary Search Tree is created by using the left to right sequence of the following item set:

{ 15, 60, 89, 99, 79, 75 }

(a) Draw the result of this Binary Search Tree. [3 marks]

(b) Show the result of the Binary Search Tree in B2 part (a) for the following traversal. [6 marks]

(i) Preorder traversal
(ii) Inorder traversal
(iii) Postorder traversal

(c) What is the height of the Binary Search Tree that you have drawn in B2 part (a)? [1 mark]

(d) Build an AVL Tree using same item set in B2 part (a). Draw the result of the insertion for the AVL Tree **step by step**. [5 marks]

(e) What is the height of the AVL Tree that you have drawn in B2 part (d)? [1 mark]

(f) Compare the searching performance of Binary Search Tree and AVL Tree. [2 marks]

(g) Convert the following expression into an expression tree. [2 marks]

(9 + 1) * 5 / 2

B3 (a) Given the following sequence of numbers:

25, 16, 8, 26, 51, 30, 22, 66

(i) Perform an **Insertion Sort** on the above sequence of numbers. Show step by step how the sorting can be achieved. The first step is shown below for your reference:

25, 16, 8, 26, 51, 30, 22, 66
[16, 25], 8, 26, 51, 30, 22, 66 [3 marks]

(ii) What is the time complexity (in Big-O notation) for a **Bubble Sort** on data in random order? [1 mark]

(iii) Perform a **Merge Sort** on the above sequence of numbers. Show your steps in partitioning and merging the list clearly. [4 marks]

(iv) Using the first number as the pivot, show how the in-place quick sort algorithm partitions the above sequence of number into two sub-lists. The first step is shown below for your reference:

25, 16, 8, 26, 51, 30, 22, 66
22, 16, 8, 26, 51, 30, 25, 66 [3 marks]

(v) Explain why **Quick Sort** is considered to be better than **Merge Sort** on data in random order. [1 mark]

(vi) Give **ONE** case that **Merge Sort** performs faster than **Quick Sort** with the first number as the pivot. [1 mark]

(vii) **Bubble Sort** is the slowest sorting algorithm on random data as the number of data swapping is much greater than other sorting algorithms. However there are some cases that **Bubble Sort** is useful and performs much quicker than the others. Describe **ONE** such case and explain what the **Bubble Sort** algorithm has to do in order to achieve the best performance in this case. [3 marks]

***** Question B3 continues in next page *****

***** Question B3 continues from previous page *****

(b) When are the time complexities (in Big-O notation) of the following algorithms?

(i)
```
int sum = 0;
for (int i=0; i<n; i++)
   sum += i;
```
[1 mark]

(ii)
```
int sum = 0;
for (int i=1; i<=n; i++)
   for (int j=1; j<=n; j++)
       sum = sum + i - j;
```
[1 mark]

(iii)
```
int sum = 0;
while (n>1) {
    sum += n;
    n = n/2;
}
```
[2 marks]

******* END OF PAPER *******