IVE Information Technology

Information & Communications Technology

Programme Board

Instructions:

(a) This paper has a total of TEN 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.

HIGHER DIPLOMA IN

SOFTWARE ENGINEERING

(IT114105)

MODULE TITLE:

DATA STRUCTURES & ALGORITHMS: CONCEPTS AND IMPLEMENTATION

MODULE CODE: **ITP4510M**

SEMESTER THREE
MAIN EXAMINATION

20 JULY, 2018
7:00 PM TO 9:00 PM (2 hours)

This paper contains TWO sections.

Section A (40 marks)

This section contains 3 questions.

Answer ALL questions.

A1 Consider the following Java code:

```
public interface Payable {
   public abstract double getPaymentAmount();
} // end interface Payable

public class Staff implements Payable {
   private String firstName;
   private String lastName;
   private String idNum;

   public Staff(String firstName, String lastName,
       String idNum) {
       this.firstName = firstName;
       this.lastName = lastName;
       this.idNum = idNum;
   }

   public String getFirstName() {return firstName;}
   public String getLastName() {return lastName;}
   public String getIDNum() {return idNum;}

   public double earnings();  //** Code Segment 1 **
   public double getPaymentAmount() {return earnings();}
} // end class Staff

public class SalariedStaff extends Staff {
   private double weeklySalary;

   public SalariedStaff(String firstName, String lastName,
       String idNum, double weeklySalary) {
       super(firstName, lastName, idNum);
       this.weeklySalary = weeklySalary;
   }

   public double getWeeklySalary() {return weeklySalary;}
   public double earnings() {return getWeeklySalary();}
} // end class SalariedStaff
```

***** Question A1 continues on next page *****

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

(a) What's wrong with the statement marked as "**Code Segment 1**" in the class **Staff**? Make corrections by modifying the headers of the class and the method "**earnings()**". (6 marks)

(b) Add a **toString** method to the class **SalariedStaff** so that the following output will be produced if the test program **PayableTest** is executed. (6 marks)

```
public class PayableTest {
   public static void main(String[] args) {
       Payable staff = new SalariedStaff("Peter", "Chan",
          "1234-5678", 5000.00);
       System.out.println(staff);
   }
}
```

Output:

```
Salaried employee: Peter Chan
ID number: 1234-5678
Weekly salary: $5,000.00
```

(c) List the **THREE** major concepts for object-oriented programming. (3 marks)

A2 Consider the following Java program.

```
public class TestException {
  public static void main(String[] args) {
    try {
      // Code Segment 2
      Circle c1 = new Circle(8);
      Circle c2 = new Circle(-5);
      c2.setRadius(-7);
      Circle c3 = new Circle(-12);
    }
    catch (InvalidValueException ex) {
      System.out.println("Exception in main thrown!");
    }
    finally {
      System.out.println("Done!");
    }
    System.out.println("Number of objects created: " +
      Circle.getNumberOfObjects());
  }
} // end class TestException


class Circle {
  private double radius;
  private static int numberOfObjects = 0;

  public Circle(double newRadius) {
    try {
      setRadius(newRadius);
      System.out.println("Circle with radius " + newRadius
                  + " set!");
      numberOfObjects++;
    }
    catch (InvalidValueException ex) {
      System.out.println("Some exception!");
    }
  }

  public void setRadius(double newRadius)
      throws InvalidValueException {
    if (newRadius >= 0)
      radius =  newRadius;
    else
      throw new InvalidValueException(newRadius);
  }
```

***** Question A2 continues on next page *****

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

```
  public static int getNumberOfObjects() {
    return numberOfObjects;
  }
} // end class Circle

class InvalidValueException extends Exception {
  private double radius;

  public InvalidValueException(double radius) {
    super(">> Invalid radius " + radius);
    this.radius = radius;
  }
} // end class InvalidValueException
```

(a) Write down the output of the above program. (5 marks)

(b) Write down the output of the program if the following segment of code replaces "Code Segment 2", surrounded by a rounded rectangle. (3 marks)

```
Circle c1 = new Circle(3);
Circle c2 = new Circle(10);
c2.setRadius(-6);
Circle c3 = new Circle(5);
```

(c) What are checked exceptions? (2 marks)

A3 Given the following linked list implementation:

```
public class LinkedList {
    private ListNode head;
    private ListNode tail;
    private int count;

    public LinkedList() { ... }
    public boolean isEmpty() { return (head==null); }
    public int getCount() { return count; }
    public void addToHead(Object item) { ... }
    public void addToTail(Object item) { ... }
    public Object removeFromHead() throws EmptyListException {
... }
    public Object removeFromTail() throws EmptyListException {
... }
    public Object getItemAt(int n) { ... }
    public Object removeItemAt(int n) { ... }
    public void addItemAt(Object item, int n) { ... }
    public String toString() { ... }
}

public class ListNode {
    public Object data;
    public ListNode next;
    public ListNode(Object data, ListNode next) {
        this.data = data;
        this.next = next;
    }
}

public class EmptyListException extends RuntimeException {
    public EmptyListException() {
        super("List is empty!");
    }
}
```

(a) Write the body for the constructor **`LinkedList()`** and the method **`addToHead()`**.
(8 marks)

(b) What is the total number of nodes in a complete binary tree with *height h*?
(2 marks)

(c) Construct a binary search tree of letters for the following sequence of input:
47 26 62 75 50 31 29 36 15 12 23
(5 marks)

Section B (60 marks)

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

B1 (a) Given the implementation of a singly linked list in Question A3. Consider the following code:

```
public interface Stack {
    public abstract void push(Object item) throws StackFullException;
    public abstract Object pop() throws StackEmptyException;
    public abstract boolean isEmpty();
    public abstract Object top() throws StackEmptyException;
    public abstract int size();
}

public class LinkedStack implements Stack {
    private LinkedList s;

    public LinkedStack() { ... }
    public int size() { ... }
    public boolean isEmpty() { ... }
    public void push(Object item) throws StackFullException { ... }
    public Object pop() throws StackEmptyException { ... }
    public Object top() throws StackEmptyException { ... }
}

public class StackEmptyException extends RuntimeException {
    public StackEmptyException() {
        super("Stack is empty!");
    }
}

public class StackFullException extends RuntimeException {
    public StackFullException() {
        super("Stack is full!");
    }
}
```

Complete the methods for `push(Object item)` and `pop()` above. (6 marks)

(b) Suggest two stack applications. (2 marks)

***** Question B1 continues on next page *****

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

(c) In addition to linked-list implementation, the stack ADT can be implemented in **`Vector`**, provided by Java API.

(i) State a similarity and a difference between linked-list implementation and Vector implementation. (4 marks)

(ii) Consider the **`Stack`** interface in Question B1 (a) and the following Java API for the class **`Vector`**. You may assume that the classes **`StackFullException`** and **`StackEmptyException`** has been given to you.

`Vector()` – Constructor Constructs an empty vector so that its internal data array has size 10 and its standard capacity increment is zero.
`boolean add(Object element)` Appends the specified element to the end of this vector.
`Object get(int index)` Returns the element at the specified position in this vector.
`boolean isEmpty()` Tests if this vector has no components.
`Object remove(int index)` Removes the element at the specified position in this vector.
`int size()` Returns the number of components in this vector.

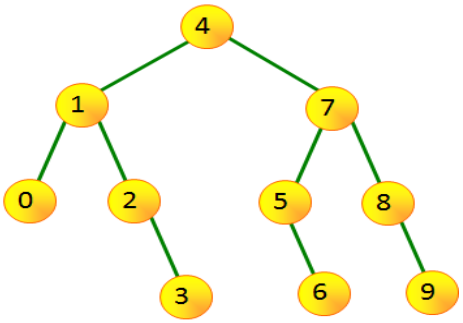
Complete the methods **`push()`**, **`pop()`**, and **`top()`** of the following implementation of the stack ADT using Java class **`Vector`**.

```
import java.util.*;

public class VectorStack implements Stack {
   private Vector vector;

   public VectorStack() {
      vector = new Vector();
   }
   public int size() {
      return vector.size();
   }
   public boolean isEmpty() {
      return vector.isEmpty();
   }

   // Method push(Object item)
   // Method Object pop()
   // Method Object top()
}
```

(8 marks)

B2 (a) In general, searching a binary search tree is much faster than a linked list. Give an example of a binary search tree with at least five nodes of numbers in which the search is in the worst case. (3 marks)

(b) Given the following binary search tree:



List the result of each of the following traversals: (6 marks)

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

(c) Complete the following missing parts for the implementation of a node of a binary search tree. (5 marks)

```
public class IntBstNode {
    private int data;
    private IntBstNode left;
    private IntBstNode right;
    public IntBstNode(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
    public int getData() {
        //(i) missing part
    }
    public IntBstNode getLeft() {
        //(ii) missing part
    }
    public IntBstNode getRight() {
        //(iii) missing part
    }
    public void setLeft(IntBstNode p) {
        //(iv) missing part
    }
    public void setRight(IntBstNode p) {
        //(v) missing part
    }
}
```

(d) Based on the implementation in Question B2(c), define a method in Java code to perform an inorder traversal of a binary search tree and print the data of each node of the tree. (6 marks)

B3 (a) Describe each of the following techniques of algorithm analysis: (4 marks)

(i) Experimental approach

(ii) Analytical approach

(b) What is the time complexity (in Big-O notation) of each of the following algorithms?

(i)
```
for (int i=0; i<n; i+=4) {
    for (int j=n; j>5; j--) {
        System.out.println(j*2);
    }
}
```
(2 marks)

(ii)
```
for (int i=0; i<=n/4; i++) {
    System.out.println(i);
}
k = n * 5 + 2;
for (int j=n; j>=1; j--) {
    System.out.println(j*2-1);
}
```
(2 marks)

(c) (i) Perform a **Bubble Sort** on the following sequence of numbers. Show the result of each of the passes. (3 marks)

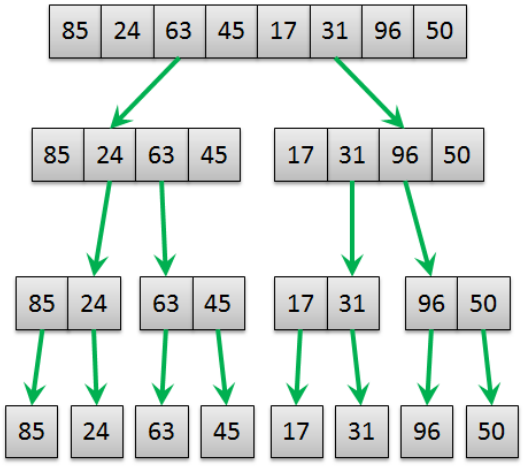
22 48 11 4 60 35 7 18

(ii) Perform an **Insertion Sort** on the following sequence of numbers. Show the result of each of the passes. (3 marks)

18 35 7 10 64 29 32

(d) (i) What is the major drawback of **Merge Sort**? (2 marks)

(ii) Given the following input sequences process at each node of a merge sort tree. Draw the output sequences generated at each node. (4 marks)



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