© Vocational Training Council

IVE Information Technology

Information & Communications Technology

Programme Board

Instructions:

(a) This paper has a total of ELEVEN 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 FOUR 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: **ITP4510**

SEMESTER TWO
MAIN EXAMINATION

10 MAY, 2018
1:30 PM TO 3:30 PM (2 hours)

© Vocational Training Council

This paper contains TWO sections.

Section A (40 marks)

This section contains 4 questions.

Answer ALL questions.

A1 (a) List what essential services are required for ***Stack*** data structure implementation by stating them in the Java interface. [4 marks]

```
public interface Stack {
        // all essential abstract methods to be provided
        public abstract int size( );
        ... ...
}
```

In each abstract method, state whether there is any return value.

(b) By applying the ***Stack*** data structure, check the parenthesis for the arithmetic expression:

[(3 + 8) * 2 + { 5 / 4] }

Copy and complete the following table on your answer book to specify the current stack content after handling each individual character. The first two steps and the last step are shown below for your reference: [6 marks]

Input character	Stack Content (after handling the character)	Return Value
[	**[**	
(	**[(**	
3		
+		
8		
)		
*****		
2		
+		
{		
5		
/		
4		
]		
}		

The parenthesis checking algorithm is given in **Appendix I**.

© Vocational Training Council

A2 An operating system developer considers applying *Stack* as the data structure to implement the process scheduler.

(a) State any ONE weakness for such an implementation. Suggest a better data structure for it.

[4 marks]

(b) Given a well-developed linked list data structure having the following methods provision:

List Methods	Description
isEmpty()	Check if the list is empty.
count()	Count and return the number of nodes.
addToHead()	Add the specified item as the first node.
addToTail()	Add the specified item as the last node.
removeFromHead()	Remove and return the first node.
removeFromTail()	Remove and return the last node.

Code the following missing methods for a *LinkedQueue* class through re-using those methods provided in the linked list. [6 marks]

(i) isEmpty (to indicate whether the queue is empty)

(ii) deQueue (to remove an item from the queue)

```
public class LinkedQueue implements Queue {

        private LinkedList  qll;

        public LinkedStack() {
              qll = new LinkedList();
        }

        public int size() {
              return qll.getCount();
        }

        public boolean isEmpty() {
              // to be completed
        }

        public Object dequeue()
              throws QueueEmptyException {
              // to be completed
        }
}
```

© Vocational Training Council

A3 The program below can be compiled successfully, however exceptions are caused during runtime.

```
1   public class CopyData {
2      public static void main(String [] args) {
3         double[] target = null;
4         int[] source = {11, 13, 4, 0, 10, 5};
5
6         for (int i=0; i<6; i++) {
7               target[i] = source[i] / source[i+1];
8         }
9      }
10  }
```

(a) State the name of the exceptions caused and explains why it happened. [6 marks]

(b) Add a try-catch block for the following program segment, such that an acknowledgment will be shown on screen when the random number is 9 and the program runs continuously without intercepted. [4 marks]

```
public class A1 {
   public static void main(String[] args) {
      int i, n;
      for (i=0; i<10; i++) {
            n = (int) (Math.random()*10);
            System.out.println(n);
      }
    }
}
```

Expected output example after modified:

```
0
0
Skipped when number is 9.
8
7
6
2
8
Skipped when number is 9.
4
```

© Vocational Training Council

A4 (a) A class **Book**, which is given in below, has implemented an interface **Lendable** containing two abstract methods: borrowItem() and returnItem().Write the interface class **Lendable**. [5 marks]

```
public class Book implements Lendable {
      private String id;
      private String title;
      private String author;
      private boolean availability = false;

      public Book(String id, String title, String author) {
            this.id = id;
            this.title = title;
            this.author = author;
      }
      public void borrowItem() { availability = false };
      public void returnItem() { availability = true };
}
```

(b) State **FIVE** problems in the following program coding. [5 marks]

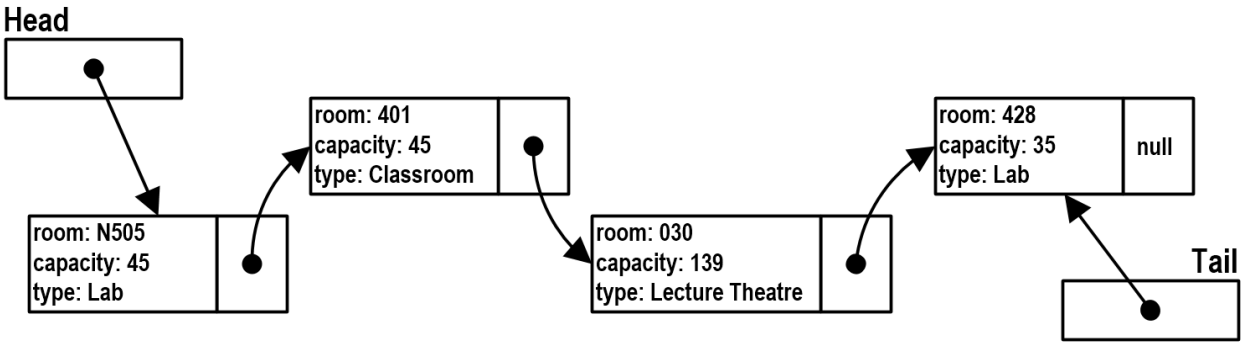
```
abstract class Weapon {
   private String name;
   private int power;
   public Weapon(String name, int power) {
     this.name = name;
     this.power = power;
     return;
   }
   public abstract int damage (int lev){
   }
}

class ShotGun implements Weapon {
   private int nofBullet;
   public ShotGun (String n, int pow, int nB) {
     super(n,pow);
     nofBullet = nB;
   }
   public int damage() {
     return power * 1.1;
   }
}

class LaserGun implements Weapon {
   private int fireRate;
   private int range;
   public LaserGun (String n, int pow) {
     super(n,pow);
   }
}
```

© Vocational Training Council

Section B (60 marks)

This section contains <u>THREE</u> questions. Each question carries 20 marks. Answer <u>ALL</u> questions.

B1 (a) Given the following diagram showing a singly linked list with four nodes. The linked list is used in IVERMS classroom management system. The node keeps room number, the number of students that can fit into the room and the type of the room. The nodes are inserted in **<u>random</u>** order.




```
public class ListNode {
    String room;  int capacity;  String type;
    public ListNode next;
    public ListNode(String room, int capacity, String type) { ... }
}

public class LinkedList {
    private ListNode head;  private ListNode tail;
    private Comparator comparator;

    public LinkedList(Comparator comparator) {
        head = null;  tail = null;  this.comparator = comparator;
    }
    public boolean isEmpty() { ... }
    public void addToHead(String room, int capacity, String type) {
          ... }

    public void addToTail(String room, int capacity, String type) {
        ListNode newNode = new ListNode(room, capacity, type);
        /*** TO BE COMPLETED in B1 part a(ii) ***/
    }

    public int campusCapacity() {
        /*** TO BE COMPLETED in B1 part a(iii) ***/
    }
}
```

(i) Describe the steps in finding all rooms with capacity bigger than 60. [4 marks]

(ii) Complete the method **addToTail ( )** of the LinkedList class. The method appends a new node to the end of the list. [4 marks]

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

© Vocational Training Council

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

(iii) Complete the method **campusCapacity** ( ) of the LinkedList class. The method reports the total capacity of the campus. The total capacity is the sum of all room capacity. [4 marks]

(b) Given the following illustration on **comparator** implementation for the IVERMS.

```
public interface Comparator {
    public abstract boolean isLessThanOrEqualTo (int item1, int
         item2);
    public abstract boolean isGreaterThanOrEqualTo (int item1, int
         item2);
}

public class IntComparator implements Comparator {
        /*** TO BE COMPLETED in B1 part b(i) ***/
}

public class LinkedList {
    private ListNode head;  private ListNode tail;
    private Comparator comparator;

    public LinkedList(Comparator comparator) {
        head = null;  tail = null;  this.comparator = comparator;
    }
    public boolean isEmpty() { ... }
    public void addToHead(String room, int capacity, String type) {
         ... }
    public void addToTail(String room, int capacity, String type) {
         ... }

    public void insertInOrder (String room, int capacity, String
         type) {
        ListNode newNode = new ListNode(room, capacity, type);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
           if (comparator.isGreaterThanOrEqualTo(head.capacity,
                 capacity)) {
              addToHead(room, capacity, type);
           /*** TO BE COMPLETED in B1 part b(ii) ***/
        }
    }

public class IVERMS {
    public static void main(String [ ] args) {
        LinkedList room = new LinkedList(new IntComparator());
        room.insertInOrder("N505",45,"Lab");
        ...
    }
}
```

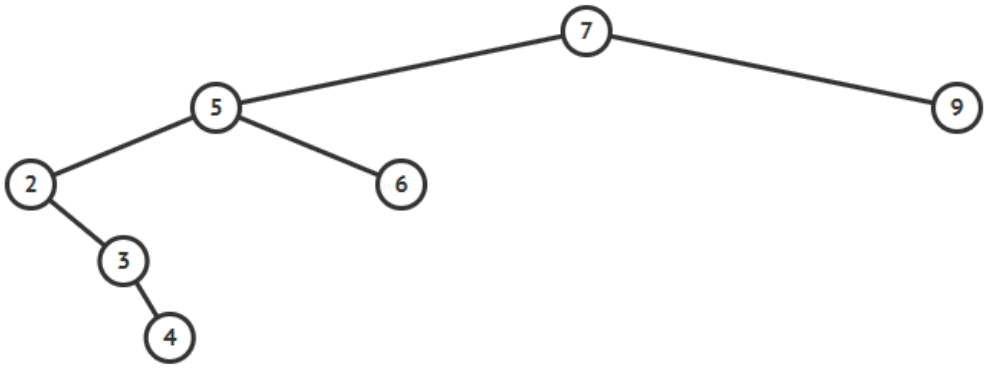
(i) Complete the **IntComparator** class. [2 marks]

(ii) Complete the method **insertInOrder()** of the LinkedList class. The method uses the **comparator** to insert the node in ascending order of capacity.[6 marks]

© Vocational Training Council

B2 (a) Build a Binary Search Tree using the numbers in the order of [31, 37, 33, 11, 5, 7]. [3 marks]

(b) Given the following Binary Search Tree.



(i) List the result of **Inorder** traversal and **Postorder** traversal from the given Binary Search Tree. [4 marks]

(ii) List the resulted array for the given Binary Search Tree that is implemented with array structure. [3 marks]

(iii) Write the result of breadth-first traversal from the given Binary Search Tree. [2 marks]

(c) Build an AVL Tree using the following item set. Draw the result of the insertion for the AVL Tree **step by step**. [8 marks]

10 , 18 , 26 , 42 , 20 , 24

© Vocational Training Council

B3 (a) Given the following sequence of numbers:

26, 38, 25, 68, 11, 30, 48, 23

(i) Perform a **Selection Sort** on the above sequence of numbers. Show the result of **each pass** of the sorting. The first pass is shown below for your reference (number(s) in [ ] is/are sorted) :

26, 38, 25, 68, 11, 30, 48, 23
[11], 38, 25, 68, 26, 30, 48, 23 [3 marks]

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

(iii) What is the best time complexity (in Big-O notation) for an **Insertion Sort**? [1 mark]

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

(v) Copy and complete the following tables on your answer book to show how the **Quicksort** algorithm (using the first number as the pivot) partitions the above sequence of numbers into two sub-lists. The first two steps are shown below for your reference:

index	**0 (pivot)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
data	26	38	25	68	11	30	48	23
Step 1	26	38						
Step 2	26	25	38					
Step 3								
Step 4								
Step 5								
Step 6								
Step 7								
Step 8 (swap pivot)								

storeIndex
1
1
2

[4 marks]

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

© Vocational Training Council

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

(vi) **Quicksort** involves the partitioning of an array into two subarrays. Explain why it is desirable that the two subarrays should have approximately the same length. [2 marks]

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

(i)
```
int sum = 0;
for (int i=1; i<=2*n+1; i++)
   sum += i;
```
[1 mark]

(ii)
```
int sum = 0;
for (int i=0; i<=n; i++)
    for (int j=n-10; j<=n; j++)
        sum = sum + i*j;
```
[2 marks]

(iii)
```
int sum = 0;
int i = 0;
while (i<=n) {
    for (int j=n; j>=i; j--)
        sum = sum + i*j;
    i+=2;
}
```
[2 marks]

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

© Vocational Training Council

Appendix I - Parenthesis Checking Algorithm

```
Algorithm   parenthesisMatching (exp)
Input:      An expression exp containing parenthesis symbols { [ ( ) ] }
Output:     matching result as a Boolean

BEGIN
        create stack S

        FOR EACH character c in exp DO

            IF c is { or [ or ( THEN
                S.push (c)
            ELSE    IF c is } or ] or ) THEN
                                = S.pop ()
                                IF c and x do not match THEN
                                    ETURN false
                                END IF
                            END IF
            END IF
        END FOR

        IF S.isEmpty () THEN
            RETURN true
        ELSE
            RETURN false
        END IF
END
```