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 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

4 May, 2023
1:30 PM TO 3:30 PM (2 hours)

This paper contains TWO sections.

Section A (40 marks)

This section contains 4 questions.

Answer ALL questions.

A1 Consider the following Java program, answer the following questions.

```
import java.util.*;
public class ChoosePartner {
  public static void main(String [ ] args) {
     Scanner sc = new Scanner(System.in);
     String [] arrayPartner = { "Kobe", "James", "Kevin", "Nash", "Rain" };
     System.out.print("Which person you will choose to be your partner?");
     try {
        int indexPartner = sc.nextInt();
        System.out.print("What is your age?");
        int age = sc.nextInt();
        /* missing code segment (d) */
        System.out.println("Your partner is " + arrayPartner[indexPartner] +
                           " ,your age is " + age);
     }catch (InputMismatchException e) {
        System.out.println("Wrong Input");
     }catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Partner not found");
     }catch (NegativeNumberException e){
        System.out.println(e.getMessage());
     }catch (Exception e){
        System.out.println("Some Exception catched");
     }finally{
        System.out.println("See you next time");
     }
  }
}
```

```
public class NegativeNumberException extends ArithmeticException{
  public NegativeNumberException(){
     super("Invalid age");
  }
}
```

State the output of the programme **after** user inputs the following values:

(a) indexPartner = **10**, age = **17**; [2 marks]

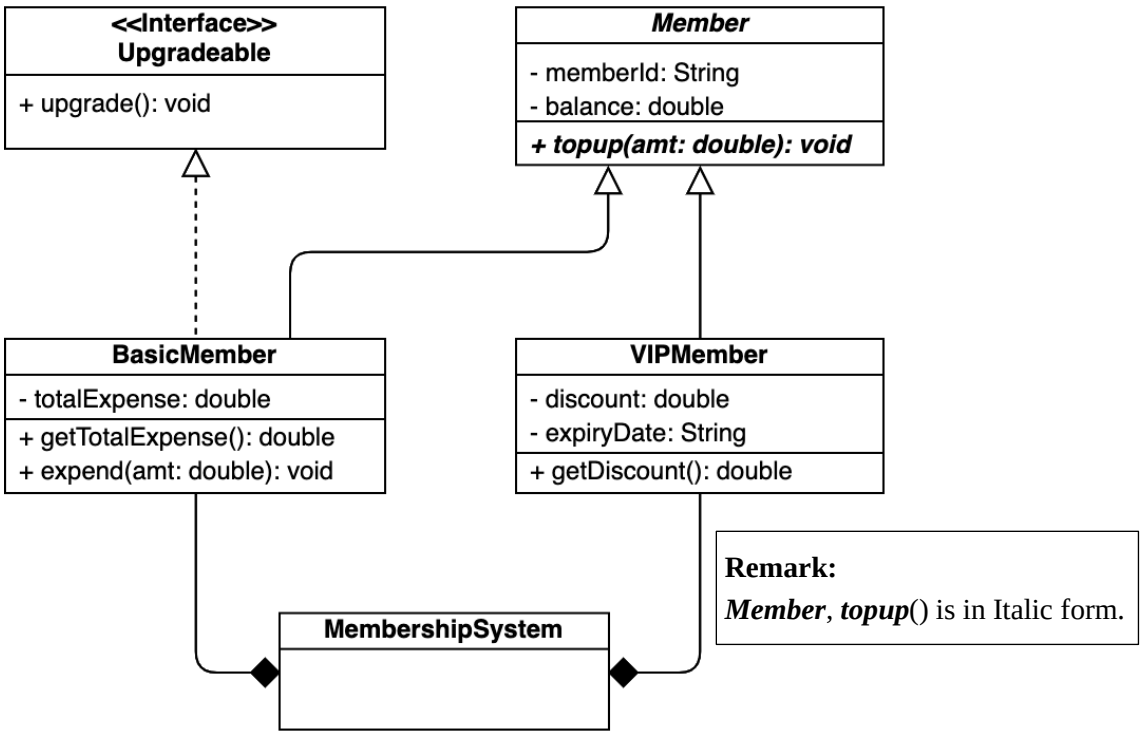
(b) indexPartner = **3**, age = **20**; [2 marks]

(c) indexPartner = **Kobe**, age = **24**. [2 marks]

(d) Complete the missing code segments to prevent user from inputting negative values (e.g. -3) as age. [2 marks]

(e) After completed the missing code segment (d), state the output of the programme after user inputs indexPartner = **2**, age = **-5**. [2 marks]

A2 Consider the interface and classes in the following UML class diagram, answer the following questions.



(a) Write the interface **Upgradeable**. [2 marks]

(b) Write the **abstract** class **Member**. [3 marks]

(c) Write the **class header** of **BasicMember**. [3 marks]

(d) Which **FOUR** methods **MUST** be implemented in the **BasicMember** class? [2 marks]

A3 Consider the following classes.

```
public class ArrayStack {
    private int capacity = 100;
    private Object[] array = new Object[capacity];
    private int top = -1;

    // size method which returns the number of element(s) in the stack
    public int size() { /* missing code segment (i) */ }

    // isEmpty method which returns true if the stack is empty,
    // otherwise returns false.
    public boolean isEmpty() { /* missing code segment (ii) */ }

    public void push(Object item) {
        if (size() < capacity) {
            array[++top] = item;
        } else {
            System.out.println("Fulled.");
        }
    }

    // pop method is to remove the data item on top of the stack
    // and returns the data.
    public Object pop() {
        if (!isEmpty()) { /* missing code segment (iii) */ }
    }

    public Object top() {
        if (!isEmpty()) { return array[top]; }
    }
}
```

The above class ArrayStack is a container of items that maintains the Last-In-First-Out (LIFO) property.

(a) Complete the missing code segments (i), (ii) and (iii) of the above ArrayStack. [5 marks]

(b) The following series of operations are performed on an empty stack one after one. Copy and complete the following table on your answer book to show the content of the given stack after each of the following operations. [3 marks]

Operations
(i) Push 400
(ii) Push 500
(iii) Pop
(iv) Pop
(v) Push 300
(vi) Top

Operations	(i)	(ii)	(iii)	(iv)	(v)	(vi)
Content						

A4 A queue can be implemented by using a circular array. The status of a circular array-based **queue Q** is given below.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Content	***null***	***null***	***null***	**Hedgehog**	**Rabbit**	**Dog**	**Cat**

(a) What is the maximum number of data that can be stored if the **queue Q** is empty? [1 mark]

(b) If the index of front is 3, what is the index of rear? [1 mark]

(c) How to check the queue whether it is empty by using the front and rear? [1 mark]

(d) Assume the queue Q has been cleared and reset ***(i.e., front = rear = 0)***. Show the final contents of the array after the following codes are executed: [7 marks]

```
for (int k = 1; k <= 6; k++)
    Q.enqueue(k);
for (int k = 1; k <= 4; k++) {
    Q.enqueue(Q.dequeue());
    Q.dequeue();
}
```

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Content							

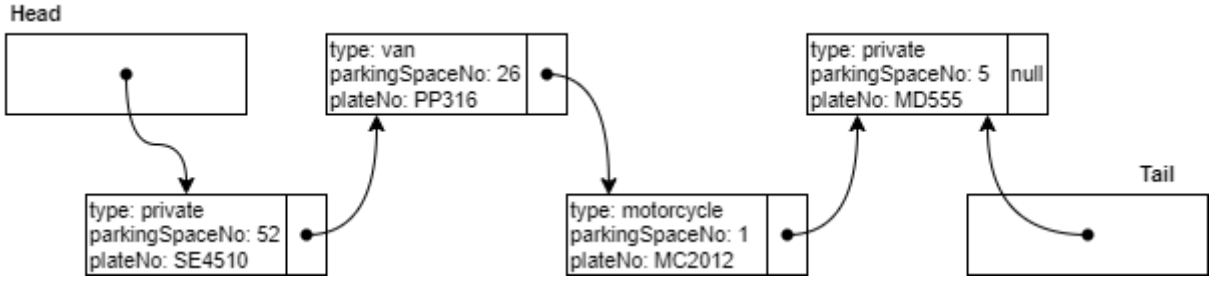
(e) A queue can also be implemented by using Linked List. State ONE advantage and ONE disadvantage of using Linked List instead of using Array. [2 marks]

Section B (60 marks)

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

Answer ALL questions.

B1 (a) The following diagram represents the Linked List used in the IVE Car Parking System (IVECPS). The node keeps the type of the car, the number of parking space and the car plate number. The nodes are inserted in **random** order.




```
public class ListNode {
    String name; int parkingSpaceNo; String plateNo; ListNode next;
    public ListNode(String type, int parkingSpaceNo,String plateNo){...}
}  // class ListNode

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

    public LinkedList() { head = null;  tail = null; }
    public boolean isEmpty() { ... }

    public void addToHead(String type, int parkingSpaceNo,String plateNo) {
      if (isEmpty())
        head = tail = new ListNode(type, parkingSpaceNo, plateNo);
      else
        head = new ListNode(type, parkingSpaceNo, plateNo, head);
    }

    public ListNode removeFromHead() throws EmptyListException {
      if (isEmpty())
        throw new EmptyListException();
      /*** TO BE COMPLETED in B1 part (a)(i) ***/
    }

    public boolean isAvailable(int parkingSpaceNo){
      /*** TO BE COMPLETED in B1 part (a)(ii) ***/  }
}  // class LinkedList

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

(i) Complete the method **removeFromHead** of the LinkedList class. The method removes the first node from the beginning of the list and returns the deleted node to the caller. [5 marks]

(ii) Complete the method **isAvailable(int parkingSpaceNo)** of the LinkedList class. The method searches for the node with the specific parkingSpaceNo and returns false when the node is found. It will return true if the node is not found. [6 marks]

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

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

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

```
public interface Comparator {
    // returns true if value of item 1 less than item 2,
    // otherwise returns false.
    public abstract boolean isLessThan (int item1, int item2);

    // returns true if value of item 1 greater than item 2,
    // otherwise returns false.
    public abstract boolean isGreaterThan (int item1, int item2);
}  //interface Comparator

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

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

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

    public void insertInOrder(String type,int parkingSpaceNo,String plateNo){
        if (isEmpty()) {
            head = tail = new ListNode (type,parkingSpaceNo,plateNo);
        } else {
            if (comparator.isGreaterThan(head.parkingSpaceNo,parkingSpaceNo)){
                addToHead(type, parkingSpaceNo, plateNo);
        } else if(comparator.isLessThan(tail.parkingSpaceNo, parkingSpaceNo)){
            addToTail(type, parkingSpaceNo, plateNo);
        } else {    // insert in the middle
            /*** TO BE COMPLETED in B1 part b(ii) ***/
            ListNode current = _______(1)_______; // locate to head node
            while (_______(2)_______) {  // check boundary
                if(_____________(3)_____________) {  // check ordering
                     _____________(4)_____________;  // create a new ListNode
                     newNode.next = current.next;
                     current.next = newNode;
                 }
                 _____________(5)_____________; // go to next node
            }
         }
     }
} // class CLinkedList
```

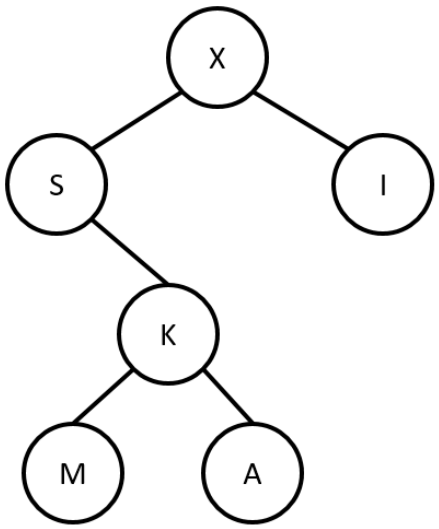
***** Question B1 continues in next page *****

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

(i) Complete the **ParkingSpaceComparator** class. [3 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]

B2 Given the following binary tree with the inserting sequence: [X, I, S, K, M, A]



(a) List the traversal result of the above binary tree. [6 marks]

(i) Pre-order traversal
(ii) Post-order traversal

(b) List the resulted array for the above binary tree that is implemented with array structure. [3 marks]

(c) Build an AVL tree using the following data set and inserting sequence. Draw the result of the insertion for the AVL tree **step by step**. [8 marks]

[5, 8, 27, 16, 11, 13]

(d) Explain why AVL tree is preferred over ordinary Binary Search Tree for data searching. [3 marks]

B3 (a) You are asked to sort the following sequence of numbers in ascending order:

33, 30, 57, 41, 74, 18, 52, 60

(i) Perform an **Insertion 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) :

33, 30, 57, 41, 74, 18, 52, 60
[30, 33], 57, 41, 74, 18, 52, 60 [3 marks]

(ii) What is the time complexity (in Big-O notation) for a **Bubble Sort** on data **in descending 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) Copy and complete the following tables on your answer book to show how the **Quick Sort** 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**	**storeIndex**
data	33	30	57	41	74	18	52	60	1
Step 1	33	30							2
Step 2	33	30	57						2
Step 3									
Step 4									
Step 5									
Step 6									
Step 7									
Step 8 (swap pivot)									

[5 marks]

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

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

(v) **Quick Sort** can use the first number or the middle number as a pivot to divide a list into two sub-lists. In some special cases choosing the middle number is better. Describe, with explanation, under what condition choosing the middle number as the pivot will result in a better performance. [2 marks]

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

(i)
```
int sum = 0;
for (int i=n+1000; i>=1; i-=10)
   sum += i;
```
[1 mark]

(ii)
```
int sum = 0;
for (int i=1; i<=n; i*=3)
        sum += i;
```
[2 marks]

(iii)
```
int sum = 0;
for (int i=1; i<=n/2; i++)
   for (int j=n; j>1; j/=3)
        sum += (i*j);
```
[2 marks]

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