

**FINAL EXAMINATION****DIPLOMA IN COMPUTER SCIENCE****COURSE : DATA STRUCTURES****COURSE CODE : SWC3684/SWC3344****DURATION : 2 HOURS AND 30 MINUTES****INSTRUCTIONS TO CANDIDATES :**

1. This question paper consists of **FOUR (4)** questions.
2. Answer ALL questions in the Answer Booklet provided.
3. Please check to make sure that this examination pack consists of:
 - i. The Question Paper
 - ii. An Answer Booklet
4. Do not bring any material into the examination hall. An electronic calculator is allowed.
5. Please write your answer using permanent ink.

MYKAD/ PASSPORT NO. : _____**ID. NO. :** _____**LECTURER :** _____**SECTION :** _____

DO NOT OPEN THIS QUESTION PAPER UNTIL YOU ARE TOLD TO DO SO

This question paper consists of 7 printed pages including the front page

QUESTION 1

- a. List **FOUR (4)** visibility identifiers used in a Java environment. (4 marks)
- b. Name the Object-Oriented Programming (OOP) concept that involves hiding internal data details and restricting external access. (1 mark)
- c. Define an Abstract Data Type (ADT). (2 marks)
- d. Identify **THREE (3)** types of errors commonly encountered in a programming environment. (3 marks)
- e. Outline the **THREE (3)** main steps required to select a data structure for a given problem. (6 marks)
- f. Identify **FOUR (4)** examples of core implementation classes provided by the Java Collections Framework (JCF). (4 marks)

(Total : 20 marks)

QUESTION 2

- a. List any **FOUR (4)** basic operations that can be performed on a linked list. (4 marks)
- b. Identify the specific variation of a linked list where the last node points back to the first node instead of pointing to null. (2 marks)

- c. Answer all the following questions based on the Java program provided below:

```
import java.util.LinkedList;

public class EditorHistoryManager {
    public static void main(String[] args) {

        LinkedList<String> history = new LinkedList<>();

        // Phase 1: Typing & Action Ingestions
        history.addLast("Type('Hello')");
        history.addLast("Type('World')");
        history.addFirst("OpenDocument");
        history.addLast("BoldText");

        // Phase 2: Targeted Document Interventions
        history.add(2, "InsertImage");
        history.add(0, "InitializeWorkspace");

        // Question 2.c.i.
        System.out.println("History Log Snapshot: " + history);

        // Phase 3: Undo Mechanics & State Validation
        // Question 2.c.ii.
        System.out.println("Undone          Command:          " +
history.removeLast());
        System.out.println("Is 'OpenDocument' still present? " +
history.contains("OpenDocument"));

        // Phase 4: Crash Recovery & Corruption Override
        int corruptionIndex = history.indexOf("InsertImage");
        history.set(corruptionIndex, "RenderGraphic");

        // Question 2.c.iii.
        System.out.println("Active History Stack Depth: " +
history.size());

        // Question 2.c.iv.
        System.out.println("Final Cleansed Log: " + history);
    }
}
```

- i. Determine the sequential contents of the `history` linked list right after Phase A and Phase B are completed.
(4 marks)
- ii. Identify the exact output text string printed to the console when the Phase 3 `removeLast()` and `contains("OpenDocument")` lines are executed.
(2 marks)

- iii. Calculate the final integer value returned by the `history.size()` expression and printed in Phase 4.
(2 marks)
- iv. Illustrate a logical diagram showing the final configuration, data elements and node links of the `history` LinkedList upon the absolute completion of the program.
(6 marks)

(Total : 20 marks)

QUESTION 3

- a. Identify the Boolean method used to verify if a stack has no elements before executing a deletion operation.
(1 mark)
- b. State **TWO (2)** common real-world applications for a stack data structure.
(2 marks)
- c. Based on your understanding of the Stack Abstract Data Type (ADT), complete the partial Java program below.

```
import java.util.Stack;

public class ElevatorFloorTracker {
    public static void main(String[] args) {

        // Instantiating a stack to store the floors visited by
        the elevator
        Stack<Integer> floorStack = new Stack<>();

        // Write your code here

    }
}
```

- i. Push **FOUR (4)** floor numbers into the `floorStack`.
(2 marks)
- ii. Print the complete original sequence of visited floors to the console.
(2 marks)
- iii. Retrieve and display the current destination floor at the top of the stack without removing it.
(2 marks)

- iv. Represent the elevator stopping and dropping off passengers at the **TWO (2)** most recent floors by removing the two top elements from the stack.
(2 marks)
 - v. Determine whether the `floorStack` is empty and display the corresponding boolean result.
(2 marks)
 - vi. Print the final remaining state of the `floorStack`.
(2 marks)
- d. Give the output of the given Java program below:

```
import java.util.*;

public class BankCounterQueue {
    public static void main(String[] args) {

        Queue<String> customerQueue = new LinkedList<>();

        customerQueue.offer("Customer_A");
        System.out.println("After offering Customer_A:" +
customerQueue);

        customerQueue.add("Customer_B");
        System.out.println("After adding Customer_B: " +
customerQueue);

        String frontCustomer = customerQueue.element();
        System.out.println("Customer at the front: " +
frontCustomer);

        String servedCustomer1 = customerQueue.remove();
        System.out.println("After first removal: " +
customerQueue);
        System.out.println("Served customer: " +
servedCustomer1);

        customerQueue.add("Customer_C");
        customerQueue.add("Customer_D");
        customerQueue.add("Customer_E");
        System.out.println("After adding customers: " +
customerQueue);

        String servedCustomer2 = customerQueue.remove();
        System.out.println("After second removal: " +
customerQueue);
    }
}
```

```

String nextCustomer = customerQueue.peek();
System.out.println("Next customer to be served: " +
nextCustomer);
String servedCustomer3 = customerQueue.poll();
System.out.println("After polling: " + customerQueue);
System.out.println("Served customer: " +
servedCustomer3);
System.out.println("Final customers in the queue:" +
customerQueue);
    }
}

```

(15 marks)

(Total : 30 marks)

QUESTION 4

- a. Answer the following questions based on the given expression.

$$- * + 15 \ 5 \ / \ 24 \ 6 \ + \ 9 \ * \ 2 \ + \ 4 \ * \ 7 \ 3$$

- i. Illustrate an expression tree.

(9 marks)

- ii. Based on your answer in **QUESTION 4.a.i.**, find the infix expression.

(2 marks)

- b. Table 1 shows a list of regional weather observation stations along with their recorded extreme temperature variance anomalies (Celsius deviation). Illustrate a Binary Search Tree (BST) showing its structural growth based on the temperature deviation values given chronologically from top to bottom.

Table 1: Meteorological Stations and Temperature Variances

Meteorological Station	Temperature Variance Deviation (°C)
Nexus-Base	+15
Polar-Alpha	-30
Oasis-West	+45
Sol-Station	+22
Frost-Ridge	-10
Dune-Tracker	+5
Peak-Delta	-40
Valley-Zulu	+18
Canyon-Rim	-5

(9 marks)

c. Define the term searching in the context of computer science. (2 marks)

d. Illustrate the insertion sort algorithm for the following numbers.

504 257 370 482 145 71 293 412 381

(8 marks)

(Total : 30 marks)

(TOTAL :100 MARKS)

END OF QUESTION PAPER

