**NANYANG TECHNOLOGICAL UNIVERSITY**

SEMESTER II EXAMINATION 2024-2025

**MH1403 – Algorithms and Computing**

May 2025 TIME ALLOWED: 2 HOURS

---

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SIX (6)** questions and comprises **FOUR (4)** printed pages.
2. Answer **ALL** questions. The marks for each question are indicated at the end of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This is an **OPEN BOOK** exam. You can bring any printed and hand-written material to the examination hall.
5. Calculators are **NOT ALLOWED** in the exam.

**Question 1.** Big-Oh Notation

(i) What is the optimal Big-Oh notation of the running time of the following function tricky(n)? (8 marks)

```
def tricky(n):
    if n <= 1:
        return 1
    return 4 * tricky(n // 2) + n
```

(ii) Given the recurrence relation $T(n) = 4T(n/2) + n^2$ and T(1) = 3, compute the optimal Big-Oh notation of $T(n)$. (7 marks)

**Question 2.** Python Program

(i) What is the output of the following code? (5 marks)

```
class Alpha:
    def __init__(self, num):
        self.num = num

    def modify(self):
        self.num = self.num + 5

def h(obj):
    obj.modify()
    obj = Alpha(obj.num)
    obj.num = obj.num - 2

a = Alpha(4)
h(a)
print(a.num)
```

**Question 2 Continues on Page 3.**

(ii) For the following code of binary search tree, what is the output of the program? Please briefly explain how you obtained your answer. (10 marks)

```
class Node:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key): # insert a new node
        if self.root is None:
            self.root = Node(key)
            return
        current = self.root
        while True:
            if key < current.key:
                if not current.left:
                    current.left = Node(key)
                    return
                current = current.left
            else:
                if not current.right:
                    current.right = Node(key)
                    return
                current = current.right

    def xyz(self):
        current = self.root
        while current.left is not None:
            current = current.left
        return current

bst = BST()
values = [43, 50, 30, 70, 63, 20, 48, 40, 37, 60, 35, 80, 26, 34]
for v in values:
    bst.insert(v)
result = bst.xyz()
print(result.key)
```

**Question 3.** AVL Tree
Starting with an empty AVL tree, insert the following sequence of numbers in order: 5, 10, 8, 4, 6, 2. You need to draw the AVL tree after inserting each number. (15 marks)

**Question 4.** Sorting

(i) We call the recursive function mergesort() to sort a list. The list to be sorted is listA = [3, 12, 6, 8, 5, 2, 9, 7]. The first recursive call is mergesort(listA). What does the second recursive call return? (5 marks)

(ii) In a Python recursive QuickSort function, the last element is always chosen as the pivot. When applying this QuickSort function to sort a special Python list containing 100,000 integers, the program crashes due to a stack overflow error. You are required to provide an example of such a special Python list that causes QuickSort function to crash. How to modify the QuickSort function to prevent this crash while keeping it recursive? (15 marks)

**Question 5.** Given two sorted Python lists L1 and L2, each containing n integers. You are required to develop and describe an algorithm to find the median of all the elements in L1 and L2 in O(log(n)) time. (20 marks)

**Question 6.** You are given a Python list G representing a grid with m rows and n columns, where G[i][j] represents the cost of passing through the element (i, j). You start at element (0,0) and must reach element (m-1, n-1). At each element (i, j), you can go to the element (i+1, j) or (i, j+1), but cannot go to element (i-1, j) or (i, j-1). To develop an **efficient** dynamic programming algorithm to find the minimum cost of the path from element (0,0) to element (m-1, n-1), you are required to define and specify the optimal substructure for this algorithm (you do not need to describe the whole algorithm). You also need to give the optimal Big-Oh notation of the running time of your algorithm. (15 marks)

**END OF PAPER**

# MH1403 ALGORITHMS & COMPUTING

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.