Student Number (Required) ______________________

Name (Optional) ________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be re-marked under any circumstances.

The test will be marked out of 50.

<table>
<thead>
<tr>
<th>Question 1</th>
<th>/10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/10</td>
</tr>
<tr>
<td>Question 3</td>
<td>/10</td>
</tr>
<tr>
<td>Question 4</td>
<td>/20</td>
</tr>
</tbody>
</table>

**TOTAL** | /50 |
General marking philosophy: a student who gives enough of an answer to show they understood what they were supposed to do, even if they couldn’t do it (or made lots of errors while doing it) should get about 50% on that question.

Full marks should be given if a solution is sound and not missing anything important.

Many of these questions can be answered in different ways – for example, there are other algorithms that could be used for Questions 3 and 4. It is not required that the students use the same algorithms as I do.

A student should only get 0/10 on a question if they made no attempt to answer it at all.
Question 1 (10 marks)

a) [4 marks] Prove that $n^3$ is not in $O(n^2)$

Hint: prove there cannot exist constants $c$ and $n_0$ such that

$$n^3 \leq c \cdot n^2 \quad \forall n \geq n_0$$

Solution: Suppose $c$ and $n_0$ exist, and $n^3 \leq c \cdot n^2 \quad \forall n \geq n_0$

$$\Rightarrow n \leq c \quad \forall n > n_0$$

But $\forall n \geq c + 1, n > c$ \hspace{1cm} CONTRADICTION

Therefore $c$ and $n_0$ do not exist, and $n^3$ is not in $O(n^2)$

Marking:

As explained under “General Marking Philosophy”, if the student shows that they understand the meaning of “big O” classification they should get 2/4 even if they don’t get anywhere with proving the result.

Students may take other approaches to proving this result. Give part or full marks based on the soundness and completeness of their argument.
b) [4 marks] Prove that \( n^2 \) is not in \( \Omega(n^3) \)

**Solution:** Suppose there exist constants \( c \) and \( n_0 \), with \( c > 0 \), such that

\[
n^2 \geq c \cdot n^3 \quad \forall n \geq n_0
\]

Then \( 1 \geq c \cdot n \quad \forall n \geq n_0 \) which gives \( \frac{1}{c} \geq n \quad \forall n \geq n_0 \)

But \( \forall n \geq \frac{1}{c} + 1 \), we see that \( n > \frac{1}{c} \) \text{ CONTRADICTION}

Therefore \( c \) and \( n_0 \) do not exist, and so \( n^2 \) is not in \( \Omega(n^3) \)

**Marking:** same as for part a)

c) [2 marks] Prove that \( n^2 \) is not in \( \Theta(n^3) \)

**Solution:** This follows directly from the proof that \( n^2 \) is not in \( \Omega(n^3) \)

**Marking:** I expect that some students will give much longer proofs. As with a) and b), give 1/2 if they show that they know the definition of \( \Theta \) even if they can't prove the result.
Question 2 (10 marks)

Suppose we have a one-dimensional array A containing 15 distinct integers in ascending order.

a) [5 marks] What is the maximum number of elements we will look at when performing Binary Search on A for an arbitrary target value x?


So the answer is 4

Marking: If the student’s answer shows that they know how binary search works on an array but they can’t go beyond that, give 2/5. If they understand how to find the answer but get it wrong, give 3/5 or 4/5 depending on how wrong they are (eg: for an answer that understands binary search but offers an answer of “8”, give 3/5. For an answer that understands binary search but offers an answer of 3, give 4/5

Students may express their answer very differently from my explanation. Some may use a decision tree – that’s fine.

b) [5 marks] If x is not present in A, what is the minimum number of elements we will look at when performing Binary Search on A for x?


So the answer is 4

Marking: basically the same as for a)
Question 3 (10 marks)

Let $S$ be a stack containing positive integer values. Write an algorithm that will return the smallest value in $S$. $S$ should be unchanged after your algorithm finishes. Your algorithm may use only the defined functions

$S$.push($x$)
$S$.pop()
$S$.is_Empty()

to manipulate $S$.

If $S$ is initially empty, your algorithm should return -1

Hint: your algorithm is allowed to create and use another stack.

Solution:

```python
def smallest(S):
    if S.is_Empty():
        return -1
    else:
        m = S.pop()
        T = new Stack
        T.push(m)
        while not S.is_Empty():
            x = S.pop()
            if x < m:
                m = x
            T.push(x)
        while not T.is_Empty():
            S.push(T.pop())
        return m
```

Marking: I'm sure there will be a great variety of answers. Please don't penalize for "syntax errors" - if the solution works, that's fine. If the solution doesn't work but it is clear that the student understands the idea of a stack and the stack operations, they should get at least 5/10.
Question 4 (20 marks)

Suppose we have a **Binary Search Tree** containing a set of n integers, some of which may be duplicates.

Write an algorithm called `count` that takes two parameters:
- `t`, which is a tree
- `x`, which is a target integer

and returns the number of vertices of the tree that contain values \( \leq x \). Your algorithm should search only as much of the tree as it needs to.

For example, if the tree `t` is

```
    7
   / \  /  \
  4   10  8  15
 /    \  \
2     7  \
```

then `count(t, 9)` should return the value 5 because `t` contains 5 vertices with values \( \leq 9 \)

You may use iteration or recursion.

Please write your answer on the next page.
Solution:

def count_LE(T, x):
    return rec_count_LE(T.root, x)

def rec_count_LE(T.root, x):
    if v == nil:
        return 0
    else:
        if v.value > x:
            return rec_count_LE(v.left_child, x)
        else:
            return 1 + rec_count_LE(v.left_child, x) + rec_count_LE(v.right_child, x)

Marking: This would be difficult to do iteratively without introducing a stack or another subsidiary data structure to keep track of vertices still be explored. However, students should not be penalized if they create a correct iterative solution. Not everyone loves recursion as much as I do. Also, I have a particular style of writing recursive algorithms: they always have an “interface” method that starts the recursion going. Students are not required to do this.

Deduct 4 marks if the student explores more of the tree than needed. In this problem that basically means that if the current vertex contains a value > x, the right child of the current vertex can be ignored.

In the textbook, binary trees are defined in such a way that each vertex has a pointer to its parent. Students should not be penalized if they use parent pointers in their solutions.

Deduct 1 mark for each minor error (such as going left instead of right if v.value < x), 2 marks for each significant error (such as not returning 0 if the tree is empty), 3 marks for each major error (such as not exploring both subtrees when
\[ v.\text{value} < x \).

Notwithstanding all deductions, a student whose answer shows they understand binary search trees and made a good attempt at answering the question should not get less than 10/20.