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</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>/10</td>
</tr>
<tr>
<td>Question 2</td>
<td>/15</td>
</tr>
<tr>
<td>Question 3</td>
<td>/10</td>
</tr>
<tr>
<td>Question 4</td>
<td>/15</td>
</tr>
<tr>
<td>TOTAL</td>
<td>/50</td>
</tr>
</tbody>
</table>

“Behind complexity, there is always simplicity to be revealed. Inside simplicity, there is always complexity to be discovered.”

~ Gang Yu
Question 1 (10 marks)

Suppose $f(n) \in \Theta(n^5)$ and $g(n) \in \Theta(n^7)$

Let $p(n) = f(n) + g(n)$

a) [4 marks] Determine the $O$ classification of $p(n)$

b) [4 marks] Determine the $\Omega$ classification of $p(n)$

c) [2 marks] If possible, determine the $\Theta$ classification of $p(n)$
Question 2 (15 marks)

Let Stack be a class that implements the stack data structure. Each instance of a Stack has three defined methods:

- push(x): add x to the top of the stack
- pop(): remove and return the top value of the stack
- isEmpty(): return true iff the stack is empty

and a single accessible attribute:

- count: the number of items currently in the stack

Let S be a stack containing integer values. Write an algorithm that will take a positive integer k as a parameter and move the bottom k values of the stack to the top. If the stack contains \( \leq k \) values, it should not be changed.

For example if the stack contains 9 and \( k = 2 \), the final result should be 3

| 3  | 14  |
| 10 | 50  |

| 9  | 10  |
| 14 | 9   |
| 50 | 8   |

Your algorithm is allowed to create and use other stacks but cannot declare arrays, linked lists or other data structures.

You may use the next page for your answer (though it should not require a full page!)
Page for answering Question 2
Question 3 (10 marks)

Here is the recursive **pre-order traversal** algorithm for binary trees.

```python
Pre_Order(v):  # v is a vertex in a binary tree
  if v == nil:
    return
  else:
    print v.value
    Pre_Order(v.left_child)
    Pre_Order(v.right_child)
    print v.value
```

Write a non-recursive version of the same algorithm (ie for a tree with a root called **root**, the given algorithm Pre_Order(root) and your algorithm your_Alg(root) must produce exactly the same output)

If you like, your algorithm may use the Stack class as defined in Question 2 of this test.
Question 4 (15 marks)

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

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

and returns the value in the tree that is closest to x. If there are two different values that tie for closeness, your algorithm should return the smaller of the two. Your algorithm should search only as much of the tree as it needs to.

For example, if the tree t is

```
    7
   / \  
  4   11
 / \   /  
2 7 8 15
```

then closest(t,5) should return the value 4 and closest(t,10) should return the value 11

You may use iteration or recursion.

Please write your answer on the next page.