NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

Question 1. [5 MARKS]

Consider the problem of finding the number of times an adjacent pair of numbers in a sequence of numbers are the same value. For example, the sequence 1 2 2 3 3 contains two adjacent pairs with the same value, and the sequence 9 9 9 9 contains three adjacent pairs with the same value.

Part (a) [3 MARKS] Give pseudocode for a Divide and Conquer algorithm to solve the problem.

Sample Solution:

```
findMatches(A, b, e):
    if b == e:
        return 0
    else:
        m = (b + e) // 2
        left = findMatches(A, b, m)
        right = findMatches(A, m+1, e)
        total = left + right
        if A[m] == A[m+1]:
            total += 1
        return total
```

Part (b) [1 MARK] Define n and give a recurrence for the worst-case runtime $T(n)$ of your algorithm.
Do NOT write a proof here.

$n = e - b + 1$

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(\frac{n}{2}) + 1, & \text{if } n > 1 \end{cases}$$

Part (c) [1 MARK] Use the Master Theorem to determine the worst-case runtime of your algorithm. (You can find the Master Theorem on page 2.)

In $T(n)$, $a = 2$, $b = 2$, $f(n) \in \Theta(1)$, so $d = 0$. Therefore $a > b^d$, so $T(n) \in \Theta(n^{log_b a}) = \Theta(n^{log_2 2}) = \Theta(n)$.

Marking Scheme:

- **Part A** [3 marks]: Full marks only for D&C algorithm that returns correct result
- 2 marks for correctly solving subproblems, but missing or incorrect combine step
- 1 mark if some attempt at dividing, but not correct
- 0 marks if no recursive calls
- 0 marks if not in pseudocode
- 0 marks on all of Q1 if not Divide & Conquer
- **Part B** [1 mark]: 1 mark if the recurrence matches the algorithm, even if algorithm is wrong. 0 marks otherwise. Any constant c in place of $+1$ is fine.
- **Part C** [1 mark]: 1 mark for correctly applying Master Theorem to recurrence in Part B. No part marks.

Marker's Comments:

- OC - algorithm overcounts
- NC - algorithm does not combine the results of recursive calls

Question 2. [8 marks]

Consider the algorithm you wrote in Question 1. Note: You may complete this part and get partial marks, even if your algorithm is not actually correct, as long as it is clear that you understand how to prove the correctness of your algorithm.

Part (a) [2 marks] Give pre- and post-conditions for your algorithm.

Sample Solution:
Pre: $b, e \in \mathbb{N}$, $b \leq e$, elements of A are comparable with each other
Post: returns the total number of matching adjacent pairs in A (or $A[b..e]$)

Marking Scheme:

- **PRecondition** [1 mark]: Full mark for precondition that matches both problem and their algorithm. Deduct 0.5 for each of: missing precondition on some parameter (missing indices count as 1), incorrect precondition for problem
- **POstcondition** [1 mark]: Postcondition comes from problem definition, so this must be correct to get the mark. Ok to say "number of pairs" here.

Part (b) [6 marks] Prove that your algorithm is correct by showing partial correctness and termination.

Sample Solution:

Inductive Step: Let $n \geq 2$.

Assume $\boldsymbol{H(n)}$: the algorithm terminates and postconditions are satisfied for all inputs of size i, $1 \leq i < n$ where the preconditions hold.

Consider a call to $findMatches(A, b, e)$ that satisfies the preconditions with $e - b + 1 = n \geq 2$.

The test on line 1 fails, so $b < e$. The next step is the computation of m, and $b \leq m < e$. Thus, both $e - (m + 1) + 1$ and $m - b + 1$ (the size of each recursive call) are less than $e - b + 1$.

By $H(n)$, both recursive calls terminate and satisfy the postconditions. That is, $left$ is the number of matches in the left half and $right$ is the number of matches in the right half. Thus, total is the number of matches in both halves.

The only adjacent pair not considered is $A[m]$, $A[m + 1]$. The total is incremented iff this pair matches. Therefore the postconditions are satisfied.

There are no additional loops or function calls, and so the algorithm terminates.

Base Case: Let $n = 1$, ie. $b = e$. Thus, the first test passes, and a return statement is executed, so the algorithm terminates. There are no pairs to consider, so 0 is correct by the postconditions.

Conclude: For all $n \geq 1$, the algorithm is correct.

Marking Scheme:

- **Size of recursive calls** [1 mark]: shows that size of recursive calls is such that they are covered by $H(n)$
- **Hypothesis** [1 mark]: applies $H(n)$ in proof
- **Base Case** [1 mark]: shows the partial correctness for the base case
- **Termination** [1 mark]: shows the algorithm terminates in all cases
- **Correctness** [2 marks]: give full 2 marks for a correct proof. Deduct 0.5 for each minor gap, 1 for each major gap (to max of 2 marks).

Marker's Comments: Note: Q1a was out of only 3 marks, although without Q2, I would consider it to be worth more. Thus, there may be deductions on Q2 as a result of errors in Q1a - the weighting of Q1a was lowered so that you would not be penalized too much for an incorrect algorithm.

You could still get full marks on Q2b even if your algorithm in Q1a was wrong as long as you acknowledged where you were not able to continue your proof due to an incorrect algorithm.

We didn't use feedback codes on Q2, you should see some specific feedback on your test paper.

Question 3. [6 MARKS]

Consider the following pseudocode to find the smallest element in a non-empty array.

```
findMin(A):
    min = MAX_POSSIBLE_VALUE
    i = 0
    # Pre: i < A.length, min = MAX_POSSIBLE_VALUE
    while i < A.length:
        if A[i] < min:
            min = A[i]
        i = i + 1
    # Post: i == A.length, min <= A[0..A.length-1]
    return min
```

Part (a) [4 MARKS] The loop invariant for the main loop of the algorithm consists of two conditions, with the first being: $i \leq A.length$. Give the second condition, which should be regarding the program variable min, and prove the loop invariant holds for all iterations of the loop.

SAMPLE SOLUTION: **LI**: $i \leq A.length$, $min \leq A[0...i-1]$.

Proof: Let $k \geq 0$ and assume $H(k) : i_k \leq A.length$ and $min_k \leq A[0..i_k - 1]$.

Case: There is no $k+1$th iteration. Then $i_{k+1} = i_k$ and $min_{k+1} = min_k$, so the loop invariant holds.

Case: There is a $k+1$th iteration, and $A[i_k] < min_k$. Then $i_k < A.length$ (since this iteration exists), so $i_{k+1} \leq A.length$ (because $i_{k+1} = i_k + 1$), and $A[i_k] = min_{k+1} < min_k \leq A[0..i_k - 1]$, by $H(n)$, so $min_{k+1} \leq A[0..i_{k+1} - 1$.

Case: There is a $k+1$th iteration, and $A[i_k] \geq min_k$. Then $i_k < A.length$ and $i_{k+1} \leq A.length$. $min_{k+1} = min_k$ so $min_{k+1} \leq A[0..i_{k+1} - 1$ by $H(n)$.

Base Case: $k = 0$. $i_0 = 0 \leq A.length$, and there are no elements in $A[0..(0-1)]$, so the second part of the LI is vacuously true.

Conclude: In all cases, the loop invariant holds.

MARKING SCHEME:

- **Loop invariant** [1 mark]: correct statement of full loop invariant
- **Inductive Hypothesis** [1 mark]: correct statement and usage of inductive hypothesis
- **All Iterations** [1 mark]: proof covers all cases for $k+1$th iteration, plus base case
- **Proof** [1 mark]: proof is correct

Part (b) [2 MARKS] Give an expression E and show that it meets the criteria to conclude that the loop terminates.

SAMPLE SOLUTION: $E = A.length - i$

1. $i, A.length \in \mathbb{N}$, and by LI, $i \leq A.length$, so $E \in \mathbb{N}$.
2. For iterations that exist, $E_{k+1} = A.length - i_{k+1} = A.length - (i_k + 1) = A.length - i_k - 1 < A.length - i_k = E_k$. So E is strictly decreasing for iterations that exist.

Therefore, by Well-Ordering, we conclude that the loop terminates.

Marking Scheme:

- **<u>N</u>atural numbers** [1 mark]: states why $E \in \mathbb{N}$
- **<u>D</u>ecreasing** [1 mark]: shows that E is strictly decreasing for existing iterations
- 0 marks for giving E without showing termination

Marker's Comments:

- PI - proof incorrect (deducted 1)
- BC - no base case (deducted 1)