Assignment 1
CPSC 331

Guransh Mangat, 30061719
Daniel Contreras, 10080311
Steven Ferguson, 30037518

Problem 1. Suppose that the function $f(n) = n$ is a bound function.

Proof. Since n is an integer input, this is an *integer-valued* function.

When the recursive function is applied in *line 9*, the value of n is reduced by at least 1 $((n-1), (n-3)$ *and* $(n-4))$.

n is a non negative integer, which means $n \geq 0$. Thus it is only possible that $f(n) = n \leq 0$ when n = 0. In that case, the test at *line 1* passes, the execution continues at *line 1* and the execution ends without having to call the recursive function repeatedly.

Hence, the function $f(n) = n$ is a bound function. ■

Problem 2. Prove that the smacG algorithm correctly solves the "MacGonagall Mystery Computation" problem.

Proof. This theorem will be proved by induction on n. The strong form of mathematical induction will be used, and base cases of $n = 0, 1, 2, 3$ will each be considered in the basis.

Basis: Suppose, first $n = 0$:
During the execution of algorithm smacG on input n, the test at line 1 succeeds and the execution continues at line 2. This causes the execution to terminate with the value of $M_0 = 1$ returned at output - as required for this case

Suppose, next, $n = 1$:
During the execution of the algorithm smacG, the test at line 1 fails and the execution continues with test at line 3.
This test succeeds, so the execution continues at line 4. This causes the execution of algorithm to terminate with value of $M_1 = 0$ returned as output- as required by this case

Suppose, $n = 2$:
During the execution of the algorithm smacG, the test at line 1 and line 3 fails, and the execution continues with test at line 5.
This test succeeds, so the execution continues with line 6. This causes the execution of the algorithm to terminate with the value of $M_2 = 5$ returned as output - as require by this case.

Suppose, n = 3:
During the execution of the algorithm smacG, the test at line 1, line 3 and line 5 fails, and the execution continues with the test at line 7.
This test succeeds, so the execution continues with line 8. This causes the execution of the algorithm to terminate with the value of $M_3 = 8$ returned as output - as required by this case.

Inductive Step: Let k be an integer such that $k \geq 3$.

Inductive Hypothesis: Suppose n is a non-negative integer such that $0 \leq n \leq k$. If the algorithm smacG is executed given n as input then this execution of the algorithm eventually

terminates, with the n^{th} MacGonagall number, M_n returned as output.

Inductive Claim: If the algorithm smacG is executed given $n = k + 1$ as input then this execution of the algorithm eventually terminates, with the $k+1^{st}$ MacGonagall number $M_{k+1} = M_n$ returned as output.

Suppose the algorithm smacG is executed with $n = k + 1$ given as input. Since k is an integer such that $k \geq 3$, n is an integer such that $n \geq 4$.
Since $n \geq 4$, the test at line 1, line 3, line 5 and line 7 fails and the algorithm continues at line 8.
Line 9 includes a recursive execution of this algorithm with the input $2 \times (n - 1)$. Since $n = k + 1 \geq 4, 0 \leq n - 1 = k \leq k$, and it follows by the **inductive hypothesis** that this recursive execution of the algorithm eventually ends with $M_{n-1} = M_k$ returned as output
Line 9 also includes a recursive execution of this algorithm with the input $-2 \times (n-3)$. Since $n = k + 1 \geq 4, 0 \leq n - 3 = k - 2 \leq k$, and it follows the **inductive hypothesis** that this recursive execution of the algorithm eventually ends with $M_{n-3} = M_{k-2}$ returned as output.
Line 9 also includes a recursive execution of algorithm with the input $n - 4$. Since $n = k + 1 \geq 4, 0 \leq n - 4 = k - 3 \leq 4$, and it follows by the **inductive hypothesis** that this recursive function of the algorithm eventually ends with $M_{n-4} = M_{k-3}$ returned as output.
Once the execution of algorithm ends, since, $k + 1 \geq 4$, it follows the definition of M_{k+1}, that the value returned as output is:

$$2 \times M_{n-1} - 2 \times M_{n-3} + M_{n-4} = 2 \times M_k - 2 \times M_{k-2} + M_{k-3} = M_{k+1}$$

as required to establish the **inductive claim.**

This results now follows by induction on n. ■

Problem 3.

Proof. ■

Problem 4.

Proof. In order to determine the number of steps included in the execution of $T_{smacG}(n)$, we will define the running time of each step in the algorithm to be *one*.

Using the uniform cost criterion to define $T_{smacG}(n)$:

The algorithm executes 2 steps (at lines 1 and 2) if it is executed when $n = 0$.
The algorithm executes 3 steps (at lines 1, 3, 4) if it is executed when $n = 1$.
The algorithm executes 4 steps (at lines 1, 3, 5, 6) if it is executed when $n = 2$.
The algorithm executes 5 steps (at lines 1, 3, 5, 7, 8) if it is executed when $n = 3$.

$$T_{smacG}(n) = \begin{cases} 2 & if n = 0, \\ 3 & if n = 1, \\ 4 & if n = 2, \\ 5 & if n = 3, \\ T_{smacG}(n-1) + T_{smacG}(n-3) + T_{smacG}(n-4) + 5 & if n \geq 4. \end{cases}$$

We can use this recurrence to show the following:

$T_{smacG}(0) = 2$ (by definition of $T_{smacG}(n)$)

$T_{smacG}(1) = 3$ (by definition of $T_{smacG}(n)$)

$T_{smacG}(2) = 4$ (by definition of $T_{smacG}(n)$)

$T_{smacG}(3) = 5$ (by definition of $T_{smacG}(n)$)

$$\begin{aligned} T_{smacG}(4) &= T_{smacG}(4-1) + T_{smacG}(4-3) + T_{smacG}(4-4) + 5 \\ &= T_{smacG}(3) + T_{smacG}(1) + T_{smacG}(0) + 5 \\ &= 5 + 3 + 2 + 5 \\ &= 15. \end{aligned}$$

$$\begin{aligned} T_{smacG}(5) &= T_{smacG}(5-1) + T_{smacG}(5-3) + T_{smacG}(5-4) + 5 \\ &= T_{smacG}(4) + T_{smacG}(2) + T_{smacG}(1) + 5 \\ &= 15 + 4 + 3 + 5 \\ &= 27. \end{aligned}$$

■

Problem 5. Suppose that T_{smacG} is a function of the non-negative integers such that, for every integer $n \geq 0$,

$$T_{smacG}(n) = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 4 & n = 2 \\ 5 & n = 3 \\ T_{smacG}(n-1) + T_{smacG}(n-3) + T_{smacG}(n-4) + 5 & n \geq 4 \end{cases}$$

then $T_{smacG}(n) \geq \left(\frac{3}{2}\right)^n$ for every non-negative integer n and therefore the number of steps executed by the *smacG* algorithm is at least exponential in its input.

Proof. This will be established by induction on n. The strong form of mathematical induction will be used and the cases $n = 0$, $n = 1$, $n = 2$ and $n = 3$ will be considered in the basis.

Basis. If $n = 0$, then $T_{smacG}(n) = T_{smacG}(0) = 2$, and

$$\begin{aligned} T_{smacG}(n) &\geq \left(\frac{3}{2}\right)^n \\ T_{smacG}(0) &\geq \left(\frac{3}{2}\right)^0 \\ 2 &\geq 1 \end{aligned}$$

as required. If $n = 1$, then $T_{smacG}(n) = T_{smacG}(1) = 3$, and

$$\begin{aligned} T_{smacG}(n) &\geq \left(\frac{3}{2}\right)^n \\ T_{smacG}(1) &\geq \left(\frac{3}{2}\right)^1 \\ 3 &\geq \frac{3}{2} \end{aligned}$$

as required. If $n = 2$, then $T_{smacG}(n) = T_{smacG}(2) = 4$, and

$$\begin{aligned} T_{smacG}(n) &\geq \left(\frac{3}{2}\right)^n \\ T_{smacG}(2) &\geq \left(\frac{3}{2}\right)^2 \\ 4 &\geq \frac{9}{4} \end{aligned}$$

as required. If $n = 3$, then $T_{smacG}(n) = T_{smacG}(3) = 5$, and

$$\begin{aligned} T_{smacG}(n) &\geq \left(\frac{3}{2}\right)^n \\ T_{smacG}(3) &\geq \left(\frac{3}{2}\right)^3 \\ 5 &\geq \frac{27}{8} \end{aligned}$$

as required.

Inductive Step. Let $k \geq 4$ be an integer. It is necessary and sufficient to use the following

Inductive Hypothesis: $T_{smacG}(m) \geq \left(\frac{3}{2}\right)^m$ for every integer m such that $0 \leq m \leq k$.

to prove the following

Inductive Claim: $T_{smacG}(k+1) \geq \left(\frac{3}{2}\right)^{k+1}$

Now, since $k + 1 > 4$,

$$\begin{aligned}
T_{smacG}(k+1) &= T_{smacG}(k) + T_{smacG}(k-2) + T_{smacG}(k-3) + 5 \\
&\geq \left(\frac{3}{2}\right)^k + \left(\frac{3}{2}\right)^{k-2} + \left(\frac{3}{2}\right)^{k-3} + 5 && \text{(by inductive hypothesis)} \\
&= \left(\frac{3}{2}\right)^{k+1} + \left(\left(\frac{3}{2}\right)^{-1} + \left(\frac{3}{2}\right)^{-3} + \left(\frac{3}{2}\right)^{-4}\right) + 5 \\
&= \left(\frac{3}{2}\right)^{k+1} \left(\frac{94}{81}\right) + 5 \\
&\geq \left(\frac{3}{2}\right)^{k+1} \left(\frac{94}{81}\right) \\
&\geq \left(\frac{3}{2}\right)^{k+1} && \text{(because } \frac{94}{81} > 1\text{)}
\end{aligned}$$

thus establishing the inductive claim, as needed. Therefore, by principle of induction $T_{smacG}(n) \geq \left(\frac{3}{2}\right)^n$ for all non-negative integers n.

■

Problem 6.

Proof. ■

Problem 7.

Proof. ■

Problem 8. By inspection of the function signature for $fmacG(n)$, this algorithm has no undocumented side effects - that is it does not access or change inputs or global data, and does not create outputs unless documented in the MacGongall Mystery Computation problem.

It now suffices to show that if the $fmacG(n)$ is executed when the precondition for the MacGongall Mystery Computation is satisfied, then either:

(a) The execution of the algorithm halts, with the postcondition for the MacGongall Mystery Computation problem being satisfied when this occurs, or alternatively

(b) The algorithm runs forever, whereby the algorithm never halts at all

Now, consider an execution of this algorithm where the precondition of this problem is initially satisfied. That is, such that n is an integer input where $n \geq 0$.

- Suppose that $n = 0$. In this case, the test at line 1 passes, and the execution of the algorithm halts after reaching line 2, where $M_0 = M_n = 1$ is returned, as required to satisfy our condition established in (a).
- Suppose that $n = 1$. In this case, the test at line 1 fails, and the execution of the algorithm proceeds and passes the test at line 3, where the execution of the algorithm halts after the execution of line 4, and $M_1 = M_n = 0$ is returned, as required to satisfy our condition established in (a).
- Suppose that $n = 2$. In this case, the test at line 1 and 3 fail, and the execution of the algorithm proceeds and passes the test at line 5, where the execution of the algorithm halts after the execution of line 6, and $M_2 = M_n = 5$ is returned, as required to satisfy our condition established in (a).
- Suppose that $n = 3$. In this case, the test at line 1, 3, and 5 fail, and the execution of the algorithm proceeds and passes the test at line 7, where the execution of the algorithm halts after the execution of line 8, and $M_3 = M_n = 8$ is returned, as required to satisfy our condition established in (a).
- The only other case is when $n \geq 4$, such that the tests at lines 1, 3, 5, and 7 fail, and the steps at $9 \rightarrow 14$ are executed, which proceeds to the while loop that is later executed. Should the while loop run indefinitely, then the execution of the algorithm never halts, and the condition established in (b) is satisfied.

  If the algorithm does not halt during loop body execution, then the loop invariant is satisfied at this point of execution, such that n is an input integer where $n \geq 4$, and i is an integer variable such that $4 \leq i \leq n$ as established in (1) and (3) of the loop invariant. Since the algorithm halts, then the test at line 15 has been evaluated and thus failed, this means $i \leq n$ as well, that is $i = n$ after the execution of the loop body.

It can now be said for part 4 of the loop invariant that $M[j] = fmacG_j = M_n$, such that the nth MacGonagal has been returned as output following the return statement at line 18 being executed, whereby condition (a) has been satisfied.

Problem 9.

Proof. ■

Problem 10.

Proof. ■

Problem 11.

Proof. ■

Problem 12.

Proof. ■

Problem 13.

Proof. ■

Problem 14.

Proof. ■