Math 416

Worksheet 1. Long division

Warning. Many of you will be familiar with *pseudocode* or the syntax of an actual programming language. While we will all soon be writing in pseudocode, be aware that some people in your group might not have seen it before! And your work in this class should never be specific to a particular programming language. For example, I don't want to see this:

```
for (int i = 0; i < n; i++) {
```

What you write should be intelligible to any other student in the class; it should never require knowledge of a particular programming language's syntax.

Practice with pseudocode and conditionals

Problem 1. Write in pseudocode an algorithm that takes as input a fruit and returns '3' if the fruit is an apple, '5' if the fruit is a blackberry, and '0' otherwise.

Problem 2. Write in pseudocode an algorithm that takes as input an integer d and returns the first 100 d^{th} powers: $1^d, 2^d, \ldots, 100^d$.

Problem 3. Write in pseudocode an algorithm that takes as input an integer N and returns the sequence of divisors of N. (You may assume that your idealized computer can check as a basic operation whether $k \mid N$.) So, for example, on input 10 the algorithm should return $\langle 10, 5, 2, 1\rangle$ (or any permutation of that sequence).

(*Hint:* Use a `while` loop.)

In gradeschool you learned an algorithm ("long division") for computing a quotient of an integer by another integer. (E.g. $416 \div 9$.) Suppose for simplicity that the divisor is a single-digit number. The purpose of this worksheet is to introduce algorithms by considering this familiar example. We will:

- give a precise description of long division; and
- prove that it always produces the correct answer.

Problem 4.

(a) Take a moment to make sure everyone in your group remembers how long division works. (It has probably been a long time since you divided numbers by hand!) Do a few examples.
(b) What is / are the input(s) of this long division algorithm? What *type* of object are they?
(c) What should the output be, in terms of the input? What does it mean for the output to be correct? (State this precisely!)
(d) What information do you need to keep track of during the course of the algorithm's execution? What *local variables* do you need?
(e) Pick one of the examples you did in part (a). Label (i.e., give names) to the input, the output, and the value of the local variables stored along the way.

Now we will try to write such an algorithm in pseudocode. [1]

Notation. Since this Long Division algorithm uses base-10 representations of numbers, it will be convenient to have notation for this.

$$(a_n a_{n-1} \cdots a_1 a_0)_{10} = a_n 10^n + a_{n-1} 10^{n-1} + \cdots + a_1 \cdot 10 + a_0.$$

Since the algorithm deals first with the *most* significant digits of the dividend, our indexing is going to be a bit backward.

[1] Your version probably used memory more efficiently, if you have some programming experience. For example, you can reuse a variable r for all of the remainders you find along the way. But this can sometimes make *analyzing* the algorithm more difficult.

Algorithm 1: Long Division

Input: the (decimal) digits $(p_0, \ldots, p_n)$ of an integer $(p_0 \cdots p_n)_{10}$ and a divisor $d \in \{1, \ldots, 9\}$.
Output: the digits $(q_0, \ldots, q_n)$ of the quotient and a remainder $r \in \{0, 1, \ldots, d-1\}$.

1 look up q_0, r_0 such that $p_0 = dq_0 + r_0$;
2 **for** i **from** 1 **to** n **do**
3 look up q, r such that $\boxed{} = dq + r$; /* $r < d$ */
4 set $q_i = q$;
5 set $r_i = r$;
6 **return** $(q_0, \ldots, q_n)$, r_n.

Problem 5. The algorithm is incomplete; what should go in the box in Line 3?

Problem 6. We should clarify Line 3. It is important that we are not asking something unreasonable of our idealized computer / 4^{th}-grader. How many pieces of information does the computer / 4^{th}-grader need to memorize (i.e., store in the *lookup table*) in order to execute any command they might encounter in Line 3? Try to give a minimal answer, which will depend on d.

Problem 7. Run Long Division to compute $416416 \div 3$. Keep track of the sequence of remainders.

(a) Using *only* the execution of the algorithm on $416416 \div 3$ (i.e., don't run it again on new inputs), deduce $4164 \div 3$ (quotient & remainder). How about $41 \div 3$?
(b) Do you see now what the first i iterations of the for loop in the algorithm achieve? This is called a *loop invariant*. State it formally by completing the following.
At the beginning of the i^{th} iteration of the `for` loop in line 2, ...

Now that you have identified the loop invariant, you must prove that it is actually a loop invariant:

Initialization: The loop invariant is true at the beginning of the first iteration of the loop.
Maintenance: If the loop invariant is true at the beginning of a loop, then it remains true at the beginning of the next loop.
Termination: The loop terminates, and at the end the invariant gives us something useful.

(So this is a proof by induction on the number of iterations.)

Problem 8.

(a) (Initialization) Why does the invariant hold at the beginning of the first iteration of the `for` loop in Line 2?
(b) What is $(p_0 \cdots p_{i-1} p_i)_{10}$ in terms of $(p_0 \cdots p_{i-1})_{10}$? (Remember that the most significant digit is the first one!)
(c) Assuming that the loop invariant holds at the beginning of the i^{th} iteration, prove that it holds at the beginning of the $(i+1)^{\text{st}}$.
(d) Complete your proof that the algorithm is correct.

Problem 9. At what point (if any!) in your proof of maintenance do you use the fact that the remainders r_i obtained by querying the lookup table are less than d? You need that the last remainder r_n is less than d for correctness, of course, but what about the others?

(*Hint:* It appears in a very sneaky place! You might find it useful to consider a specific example, perhaps the one from Problem 7.)