

## The Computational World

### Problem Set 2: Week 2

**Problem 1. (4 points)** Read Jeannette Wing's article on "Computational Thinking", and write a brief (~300 word) response to it. As an example, you might want to specify an area of your own interest in which "computational thinking" (beyond programming alone) has played a role; or you might outline an area in which students can be taught to think more algorithmically than they have been taught to do in the past; or perhaps you might disagree with Wing altogether, and argue that "computational thinking" is not as important as she thinks.

### Problem 2. 8 points

2a. Write out the values of  $F1(0)$  through  $F1(8)$  for the following function. What does this function do?

```
Define F1(N)
  IF N = 0
    RETURN 1
  ELSE RETURN F1(N-1) + F1(N-1)
```

2b. Write out the values of  $F2(1)$  through  $F2(6)$  for the following function. What does this function do (can you express what's going on here in a brief, informative way)?

```
Define F2(N)
  IF N = 1
    RETURN 2
  ELSE RETURN [F2(N-1)]2
```

2c. Write out the values of  $F3(1)$  through  $F3(6)$  for the following function. What does this one do?:

```
Define F3(N)
  IF N = 1
    RETURN 2
  ELSE RETURN 2[F3(N-1)]
```

2d. Write out the values of  $F4(0)$  through  $F4(8)$  for the following function. What does this function do? Can you express the value that it seems to be approaching?

```
Define F4(N)
  IF N = 0
    RETURN 1
  ELSE RETURN SQRT(1 + F4(N-1))
```

### Problem 3. 6 points

Here's a recursive algorithm (actually, an ancient algorithm, due to Euclid), for finding the greatest common divisor (GCD) of two non-negative integers. In the following definition, we assume that  $M$  is greater than or equal to  $N$ :

```
Define GCD (m, n)
  IF N = 0
    RETURN M
  ELSE RETURN GCD (N, Remainder(M, N))
```

The meaning of "Remainder ( $M, N$ )" is the remainder that we get when we divide  $M$  by  $N$ . (If  $N$  divides into  $M$ , then the remainder is zero.)

What is the overall meaning of this recursive algorithm? The idea here is that in order to find the GCD of two numbers, we have two possibilities: either the second number is 0 (in which case the answer is easy), or we can find the answer to our problem by calling the GCD function again on  $N$  (as the first argument, this time) and the remainder of  $M$  divided by  $N$ .

Here's an example of the algorithm in action. Suppose we want to find the GCD of 100 and 6. The result of calling GCD (100, 6) gives us the following sequence of recursive calls to the GCD function:

```
GCD (100, 6)
GCD (6, 4)
GCD (4, 2)
GCD (2, 0)
```

and when we finally call the GCD function on 2 and 0, the answer is 2. So the greatest common divisor of 100 and 6 is 2.

Show the sequence of recursive calls to GCD that give you the answers to the following problems:

```
Greatest common divisor of 100 and 8
Greatest common divisor of 100 and 7
Greatest common divisor of 101 and 8
```

### Problem 4. 7 points

Here's a small (four-rule) context-free grammar for a subset of English:

```
S -> NP VP
NP -> Det N PP*
VP -> V NP PP*
PP -> Prep NP
```

Using this grammar, show four distinct "parse trees" for the sentence:

*The man saw the woman with a telescope from the store.*

For extra credit, show a fifth parse tree for the sentence.