Don Petersen

S6 Discussion Questions

1) What is recursion? What is indirect recursion?

Recursion is when a method calls itself. Each time the method calls itself creates a new environment in which to work. All local variables and parameters are newly defined with their own unique data space. Recursion can happen directly or indirectly. When a method calls itself, it is a direct recursion. But, when a method calls a different method and then the second method calls the first method this is also recursion and is called indirect recursion. (Lewis, 2007)

2) What is base case? How do the base cases relate to infinite recursion?

Recursion is when a method calls itself. Each time the method calls itself creates a new environment in which to work. All local variables and parameters are newly defined with their own unique data space. In order to get the method to stop calling itself another option is needed. The option(s) that keep the method from calling itself is called the base case and without it the method will continue to call itself indefinitely, when this happens it is referred to as infinite recursion. (Lewis, 2007)

3) What are the similarities and differences between repetition and recursion? How have you used recursion in your prior programming experiences. Explain.

Repetition or iteration is similar to recursion in that an operation(s) are called over and over again until a desired result occurs. The iterative method accomplishes this through the use of a loop where the conditions of the loop are defined. Recursion accomplishes a series of repeated operations through calling itself over and over until a condition called a base case occurs. Recursion uses if-else statements where most commonly the base case appears in the if portion of the statement. Off the top of my head I cannot think of an example of when I have used recursion in programming. I, so far, have relied on the iterative approach to solving problems because of it being more straightforward. I am curious to see how I might be able to learn to “think recursively,” which seems to be the key to being to use recursion in programming according to the textbook. The two examples in Chp11 give good examples of the use of recursion but weren’t particularly easy to for me to follow, let alone try and duplicate on my own. (Lewis, 2007)

4) When should you use recursion? When should you not use recursion? If you do not use recursion appropriately, what are the consequences?

Recursion should be used when it will provide a more intuitive solution than an iterative one. “All problems can be solved in an iterative manner, but for some cases the iterative version is much more complicated.” (Lewis pg598) Indirect recursion appears to cause a particular issue with readability and requires extra care and clear documentation. Therefore, should only be used when truly necessary. An example in the textbook shows both the recursive and iterative methods for solving a sum of numbers between 1 and N. It also describes that in this case the recursive method is much slower and complex than the iterative one. These are the consequences of choosing a recursive approach over a iterative one, the program might be harder to understand because of its complexity and might also be slower than it would have be if another approach was used. (Lewis, 2007)

5) Name two things you would like to see added to this course that would make it a better course.

The first addition I think that I would like to see in this class is something that I say should be added to all classes online that Baker offers, video lectures and tutorials. It is incredibly helpful to see another person actually work through a problem and explain each step they took and why. The second addition relates to the first but more generally and that is more instruction from the course. I think the course does a great job of providing tasks for us to accomplish but as far as actual instruction of how to complete those tasks we are mostly left to our own devices with only the textbook material as a resource. Now, Mr. Dunn has been the most excellent teacher for making the effort to explain things step by step on a one on one basis if a student were to email him. But my experience so far is that he is exceptionally responsive to this form of interaction and it isn’t always the case with a different teacher. I think that the type of one on one explanation I received from Mr. Dunn could be adapted into more general lessons or lectures for the entire class.

6) Name two things you did not like about this course and explain.

I don’t have much to say about what I didn’t enjoy about the course that wouldn’t be a repetition of the previous questions answer. I guess if I were to name something I didn’t like, it would be that the course seemed to assume that you have taken previous courses in a programming language or had some sort of background experience but there are no pre-requisites for this course listed anywhere I have seen. This is me being nit-picky, since I am struggling to find much that I didn’t like, other than feeling completely unprepared for the difficulty of Seminar 5’s programming assignment.

Lewis, J. (2007). C# Software Solutions: Foundations of Program Design.Boston, MA:Pearson Education.