Aidan Leuck

HW6

8.20

A) No I don’t think that there is any reason that garbage collected languages should allow the user to manually de-allocate memory. I think the main appeal to languages that use garbage collection are that they automatically deal with memory for you. There are other languages such as C++ that allow you to do get close to the memory and hardware. If you are writing a program that requires precise memory deallocation, you should use a language that allows you to do so. Furthermore, garbage collection is pretty efficient and in most cases more efficient than the programmer itself. I understand there may be certain cases where it might be useful, but the performance hit of garbage collection isn’t as much of a concern with modern systems.

B) No, I think tenuring an object sounds like an even worse solution than the previous solution. This is how memory leaks, and errors happen, the programmer tenures an object and then forgets to deallocate the object when he is done. There are countless examples where a programmer forgot to deallocate memory and it ended up costing peoples lives. I think what makes this an even worse idea is that it may get overlooked when doing code reviews. Say for example a program was written in Java and a programmer tenured an object and then forgot to deallocate it. I could see the error getting easily overlooked even by those tasked with reviewing the code because they expect Java to handle deallocation. In general if a programming language has automatic garbage collection, only allow automatic garbage collection. I believe not picking a stance will lead to problems.