1. Languages are not equal because the semantics affects how we think as programmers. For instance, a python (Von Nuomen) vs a schme (functional) programmer will think differently in terms of how to solve the problem because the languages use different paradigms. However, languages can solve the same problem, usually with just a different way of thinking about it. Thus, that makes all the difference in terms of computational complexity and readability. In a language like Clisp, many would say that it maps well from idea to program syntax, which makes it simpler for development.
2. The top down approach takes and tries to solve the whole problem in whole swoop. It begins with the big picture and works down into small segments. The bottom-up approach takes the problem and breaks it into small pieces from the beginning so that sub-programs emerge. Top down requires too much detail and prevents a programmer from winging it. It’s similar to the waterfall method, which is ineffective when requirements are not well understood. The bottom-up method is good when modules are just required to get working in order to build the whole system. It allows programmers to think smaller module based problems which helps reusability.
3. Alan Turning would say that if a user has a machine and a human can’t tell the difference, then that would consist as intelligence.
4. I think humans are intelligent. My definition of intelligence is something that can reason effectively.
5. The Turning test was basically where a human would be in a room. The human would talk to a computer prompter and it could be a human or it could be a computer. If the human at the prompter couldn’t tell the difference between the human and the computer, then it must be intelligent.
6. Not Sure
7. Both are logical expert systems. When you go from data to the decision, it’s forward chaining (rule driven). When you go from the decision to data, it’s backwards chaining (goal driven).
8. The two main search types are breadth first search and depth first search. BFS uses a queue by going through each of the nearest neighbors and constantly adding them. It dequeues one and continues the pattern. DFS uses a stack by going to each of the nearest neighbors and constantly pops the last one so explores as far as possible before backtracking. DFS is good for maze generation. BFS is good for finding the shortest path.
9. Not Sure
10. Yes, because a relative intelligence ordering system could be smart in its given domain. It may not know how to order something, but it could order something else good.
11. Psychologists use have a large argument between nature and nurture. The question is whether nature or nurture impacts human learning more – is it driven by genes or upbringing. It’s applicable to AI because when trying to create and AI, humans can try to program as a much as possible into its initial reasoning capabilities, but to what extent must it also be able to change that and learn over time.
12. Weak AI – focused on a narrow task; Strong AI – as skillful as a human. Weak AI because a lot more ground can be broken in Weak AI right now.
13. Not Sure
14. I would evaluate its success based on if it can traverse the house without getting stuck. It’s really difficult for robotics to maneuver, and as long as it can get through the house, the rest (cleaning, fetching things) would be easy.