You have 60 minutes.

Show your work and explain your answers. Good work often earns partial credit. A correct answer with no explanation often earns little or no credit.

If you are asked to write code but you do not know the exact Python required, then try to write code that is approximately correct. If you think that your code does not demonstrate that you understand the solution, then describe your idea in English as well. Be precise enough that I cannot misinterpret your solution.

Good luck.

Compare our various sorting algorithms. Which is best?

Compare the performance of `__setitem__` for our various implementations of `Dictionary`.

The following graph shows seven computers in a network. Each network link is bidirectional and labeled with its latency (the time cost of making contact across that link). Execute Dijkstra's algorithm on the network completely, so that the shortest path from computer D to computer G is found. Your work must clearly show how the results-so-far and the queue evolve over time.

Below is a list of numbers. Perform heap sort on this list. For grading purposes, on each intermediate step you must display the heap as a list.
[17, 32, 21, 20, 15, 35, 12, 15]

Give an example of a problem that is better solved using depth-first search than using breadth-first search.

Recall that a scene tree is a tree of drawable objects, such that each object is positioned relative to its parent. That is, a parent sets up its coordinate system before telling its children to draw; each child's drawing automatically happens in whatever coordinate system is current. Our main example was a dog regarded as a tree of body parts. In reality, scene trees are usually called *scene graphs*, because they are graphs not trees. Why? What extra capability do graphs offer, that might be useful for organizing drawable objects in a scene?