You have 150 minutes.

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

When asked to "explain" something, explain with as much precision and detail as possible.

When asked to describe the "running time" or "efficiency" of an algorithm or data structure, state your final answer in big-$\mathcal{O}$ notation.

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.

If you have no idea how to solve a problem, or if you have forgotten a key concept that you think you need to know, you may ask me for a hint. The hint will cost you some points (to be decided by me as I grade your paper), but it may help you earn more points overall.

Good luck.

1. Give an example of how priority queues have been used in service to other data structures or algorithms in this course. Then give a second example, of how a priority queue might be applied to a real-world situation.

2. What is undesirable about `BinarySearchTree`s? How is this problem solved?

3. In this question, assume that your reader understands the basic heap operations of insertion and deletion. You do not need to explain how they work.

A. The heap data structure possesses a `buildHeap()` operation that constructs an entire heap from a list of n objects. Describe this algorithm and its running time. (You do not need a rigorous proof of running time, but your explanation should give as much intuition as possible.)

B. Describe the heap sort algorithm and its running time.

4. Implement `Queue` using a linked list (not a Python list) so that both `enqueue()` and `dequeue()` are $\mathcal{O}(1)$. (You can either write Python code or explain in words and pictures; either way, be detailed and precise.)

5. In as much detail as possible, analyze the running time of quick sort. Include both best-case and worst-case scenarios.

6. Write a recursive function to evaluate a parse tree. (Assume that the value stored in each tree node is a string. Do not assume that the tree is binary.)

7. Both breadth-first search and Dijkstra's algorithm are used to find shortest paths in graphs. So what is the difference between them? In what situations would you use one or the other?

8. A. Draw the graph corresponding to the maze drawn below (so that we can talk about solving the maze as a graph search problem, on the next page).

B. Which would work better for solving such maze problems: breadth-first search or depth-first search? Do both work? For large mazes, how do their running times compare? How do their space (memory) requirements compare? How do the quality of the solutions compare?

9. A. Suppose that we have a hash table of capacity k, with n objects in it. Collisions are resolved by chaining. Describe the best- and worst-case running time of the `get()` and `put()` operations.

B. Why did we design our `HashTable` data structure to resize itself automatically? Include a detailed best-case running time analysis.