**Project 2**

**Artificial Intelligence**

**CSCE 5210 – Fall 2021**

**Distributed: Tuesday, September 28**

**Due: Tuesday October 19**

**Part A**

Q1) From the map above what data structure needs to be derived to support the minimization of movement between divisions that is required to support consecutive orders that involve different divisions?

Answer:

* Order data structure:
* division: The division goal
* shelf: All shelves that need to be visited
* Node data structure:
* child nodes (left and right)
* parent node
* level
* path cost from the parent
* number
* Tree data structure
  + Name
  + Root: The first node
  + Tail: The last node
  + Current\_node

Q2) How will we populate the data structure that you proposed in Q1 above? Outline the procedure involved. You only need to describe the procedure, not implement it at this stage.

Answer:

* By observation, we can tell that child node’s number are equal to (their parent’s \* 2) + 0 / + 1 and parent node’s of a node is equal to floor(the node / 2)
  + For example: node 5 has children at 5 \* 2 + 0 and 5 \* 2 + 1, which are 10 and 11. Similarly, node 12 or 13 has a parent at floor(12 / 2) or floor(13 / 2), which is 6.
* When a new node is added, all we need is only the path cost from its parent. The tree will automatically find the correct position and give the node the correct id.
  + Id and position of the added node is based on the tree’s tail. For example, if the tail’s number is 12, the next node’s number is 13, and the node is traversed according to this path: 13 <- 6 <- 3 <- 1. Because 13 is odd, it is the right child of 6.
* Because of the exponentially added complication from creating subtrees. We use two trees: one for the map of divisions and the other for multiple maps of shelve locations.

**Part B**

Q3) Implement Q2 above by writing a Python program that populates the data structure you proposed in Q1.

Please see the code

Q4 Implement Iterative Deepening Search (IDS) to locate shelf locations for servicing customer orders.

Please see the code

Q5) Test your program for this boundary case. Generate an order for division 6 and trace its path from division 1 to division 6 and then onwards to the only item, item 33, that was ordered. Check that its path length is what you expected. Print out both the path and its length.

Please see the code

Q6) Run your program for a certain number N of customer orders (say 100) and answer the following questions:

1. What is the average path length travelled across the N orders? **Take care to include in your path length both the paths travelled to get to the division that contains the order and the movement required within the division to get to the items required.**

Please see the code

1. Print out the length of the shortest and longest paths across the N orders that you generated.

Please see the code