CS 370 Snake Solver – Pseudo code

# Solution requirements

1. Don’t hit the body or walls
2. Always have a path to the tail
3. Avoid creating holes
4. Leave minimal open space

# Methods

/\* Returns true if there are no obstructions between the head and apple, and false otherwise. It could simply check the direct path rather than the best path.

\*/

hasOpenPath(sx, sy, ax, ay) : Boolean

//todo

/\* Returns the shortest path from the head to the apple. It should be used when there are no obstructions to the apple.

\*/

shortestPath(sx, sy, ax, ay) : List<Commands>

//todo

/\* Returns true if the snake can navigate to any given coordinate, and false otherwise. This is used to check if the head is able to reach an apple or the tail.

\*/

hasPathTo(sx, sy, x, y) : Boolean

//todo

/\* Returns True if the head at the given xy will have a path to the tail’s xy, otherwise returns False.

\*/

canReachTail(sx, sy, tx, ty) : Boolean

//todo

/\* Given the position of the head and a list of the relevant body components, generate a new path around the body to the apple.

\*/

pathAroundBody(hx, hy, ax, ay, List<Positions>) : List<Commands>

//todo

/\* Implementation is uncertain. Should have a function which remembers the path the tail takes and translates those to commands for the snake to follow. This is used to avoid shortening the distance between the head and tail when navigating very windy paths, as the head is built to naturally take shortest paths and would catch up in certain circumstances.

\*/

followTail(?) : List<Commands>

//todo