COSC 370 – Artificial Intelligence
Project 1

Purpose: Do a little bit of agent design and create an agent to "Hunt the Wumpus".

Task: "Hunt the Wumpus" was a very early videogame, originally created by Gregory Yob in the early 1970's. In this game, your task is to guide a character to retrieve the gold from a dungeon/cave/dark place while avoiding pits and the eponymous Wumpus. The character needs to return to the starting spot to escape the dungeon. The dungeon is laid out in our version as an n by n grid (the n is chosen by the calling driver and not revealed to the agent). The trouble is that the character can't really see anything and must rely on a set of sensory inputs to get to the gold and back out of the dungeon:

- Stench - the Wumpus is in a directly adjacent square (not diagonal).
- Breeze - there is a pit in a directly adjacent square (not diagonal).
- Glitter - the gold is in the current square
- Bump - you walked in to a wall of the dungeon
- Scream - the Wumpus was killed!

The character has a limited amount of moves available:

- Move north, south, east, or west. We assume that increasing numbers along the rows indicates a move south and increasing numbers along the columns indicates a move east. So, grid[0][0] would be our top left corner, grid [1][0] would be south from that spot.
- Shoot north, south, east, or west. The character shoots in the appropriate direction. The arrow keeps going until it hits a wall, or a Wumpus. If it hits the Wumpus, the Wumpus dies.
- Grab the gold. The character grabs the gold if the gold is in the current space.
- Climb. The character climbs out of the dungeon if he or she is in the starting space.

Your goal is to create an intelligent agent that will guide our character through the Wumpus World and be as successful as possible. To do this, you will leverage the state-based agent techniques in Chapter 2 and 3 of your book. You will also be working with a provided driver that will simulate our Wumpus World. Your python code will need to interface directly with this driver (available soon!) through the following means:

Your agent filename: WumpusAgent.py
Parameter setter: setParams(type, arrows, wumpi)
Main Function: getMove(sensor)

The simulator/driver will send sensory information (see above) to your agent and the agent will need to return a move. This will continue until either the agent has succeeded in getting the gold out of the dungeon, or the agent has died. The driver will allow the tester (you) to choose a type of game (including number of arrows, and number of wumpi) and the number of iterations to run through. It will output the number of agent successes, agent failures, and the percentages.

Parameter Setter Information: the parameter setter for the agent module should take three parameters:

type - 1 for non-moving wumpi, 2 for moving wumpi
arrows - number of arrows
wumpi - number of wumpi

Note, you should catch any errors in input as part of your parameter setter function. You should default to 0, 1, 1 for the parameters respectively.

Main Function Information: the main function should take in a String that has a series of letters, denoting what sensory input detected by the character:

S - stench
B - breeze
G - glitter
U - bump
C - scream

These characters can appear in any order in the string, but will only appear at most once per string, regardless of the number of "instances" of that sensory input (for instance: if there are two pits in adjacent spaces, you wouldn't receive two B inputs).

The main function should return a String denoting the move that the agent has decided for the character:

N - move north
S - move south
E - move east
W - move west
SN - shoot north
SS - shoot south
SE - shoot east
SW - shoot west
G - grab gold
C - climb out

You are required to work in teams of 3-4 for this project. Team requests are due by 5pm, Friday, January 29th. If you do not have a team request in at this point, you will be assigned a random partner.

Learning Targets: intelligent agent design and implementation

DUE: February 18th at 11:59pm via Blackboard. Question cutoff is February 18th at 4pm.

Potential deductions and other grading remarks:

-100 does not run with the driver or does not run to completion
-15 creates an infinite loop via behavior (for instance, you are on one side of a pit wall, all wumpi and the gold are on the other, your agent should deal with this case)
-10 per each percept not handled in some way. If you choose to ignore a percept you must note in the comments a reason why.
-15 per each unrecognized command from the agent
-10 not handling input errors
-10 lack of comments, including header

Your header comment must include:
Team member names
File name
Date
Assignment
Short description

Two more grading notes: attempting to access the simulation variables directly while the simulation is running to learn the location of the gold, wumpi, pits, etc., is expressly forbidden and will result in a 0 for the project. Also, performance will be a factor!

Extra credit (15 points): build a python based graphical driver for the Wumpus World simulation. It needs to have a grid that shows the location of all elements using a visual (non-string) representation. It should also have inputs on the GUI that allows the user to adjust the grid size, number of arrows, number of wumpi, and type of wumpi.