Models and Simulations: Problem Set 4

Instructions: This week, you will write your first agent-based model (ABM). By the end of this problem set, you will have programmed the model described on pages 122-125 of Alexander's *The Structural Evolution of Morality* and Skyrms (2003) pages ???. In the model, agents are located on a grid, and each agent is assigned a random strategy (i.e., "Stag" or "Hare") on the first stage of play. On each stage, each agent plays a Stag Hunt with her nearest eight neighbors on the grid. The agent then updates her strategy using the "Imitate the best" rule described on pages 39-40 of Alexander's *The Structural Evolution of Morality*.

On the course website, you can find a Netlogo file called, "Stag Hunt" that contains part of the model. Some of the procedures in the code, however, are missing. The first three exercises below asks you to write one of the missing procedures for the program. When you are finished, you should run the model several times (by clicking the "go" button) and convince yourself that stag-hunting emerges under a wide variety of circumstance.

Exercise 1: Initializing Strategies:
In the Interface, create a slider called Percent-Stag-Hunters. Choose a minimum value of 0, a maximum value of 1, and an increment of .01. Now move to the code tab. Fill in the program called set-strategy so that it does the following. The procedure set-strategy randomly changes each patch's strategy variable to either the string "Stag" or the string "Hare", and the probability that it is assigned "Stag" is equal to the value of the slider Percent-Stag-Hunters. That is, if Percent-Stag-Hunters is .5, then there is a 50% chance that a random patch will be a stag hunter and approximately half the patches will hunt stag. If Percent Stag Hunters is .1, then there is a 10% a patch will be a stag hunter, and approximately one in ten patches will hunt stag.

Click the setup button to see if your program works. If you move Percent-Stag-Hunters towards zero, after clicking setup, more patches ought to be white. Conversely, If you move Percent-Stag-Hunters towards one, after clicking setup, more patches ought to be black.

Hint: If you apply the random-float function (see Netlogo dictionary) to the number 1, it will report a random decimal number between 0 and 1. Moreover, the probability that random-float 1 returns a number less than x

is precisely x. For instance, random-float 1 returns a number less than .1 with probability 10%.

Exercise 2: Playing the Game:
Write the procedure play-neighbors so that does the following. Each time the procedure is called, each patch plays the stag hunt game eight times. In particular, each patch plays the game once with each of its eight neighbors. It then stores the payoff of each of the eight games in the list payoffs. Use the following matrix to assign the payoffs.

	Stag	Hare
Stag	3, 3	0, 2
Hare	2, 0	2, 2

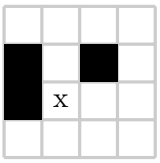
Hint: Recall that, for each patch, there is an agent-set called neighbors. So it might help to begin your code as shown below. Doing so allows you to ask each patch to play the game with its eight neighbors. Fill in the code marked with "..." so that, depending upon the strategy of the neighbor and the value of mystrategy, the appropriate payoff is added to the end of the list new-payoffs.

```
to play-neighbors
  ask patches
  [
    let mystrategy strategy
    let new-payoffs n-values 0 [?]
    ask neighbors
    [
    ...
    ]
   set payoffs new-payoffs
  ]
end
```

Exercise 3: Imitation:
Write the procedure switch-strategy so that it does the following. Each patch first determines which of its neighbors received the greatest payoff on the previous stage. It then mimics the strategy of one of its neighbors that received the greatest payoff.

For example, consider the patch marked "x" in the grid below. Stag hunters are indicated in black, and hare hunters are colored white. Suppose

the payoffs list of each of the neighbors of patch x are as below. Then the patch has two neighbors that have received a payoff of 3, both of whom are stag hunters. So patch x changes its strategy to "Stag".



1. [0 0 0 0 0 0 0 0]
2. [2 2 2 2 2 2 2 2]
3. [2 2 2 2 2 2 2 2]
4. [2 2 2 2 2 2 2 2]
5. [2 2 2 2 2 2 2 2]
6. [2 2 2 2 2 2 2 2]
7. [3 0 0 0 0 0 0 0]
8. [0 3 0 0 0 0 0 0]

Exercise 4: Stopping the Model:
Explain what the stop? reporter does. Predict what will happen if the following code is removed from the go procedure. Remove the code and the click the go-button. What happens?

```
 if stop?
[
   stop
]
```