DO NOT WRITE ON THIS QUESTION SHEET.
This test is out of 45 points, with an additional 5 points extra credit.

1. (5p) Create a reporter called *IsEven* which will be given an integer as an argument and will report True if the number is even, otherwise report False.

2. (5p) Create a procedure called *VertStripes* that will paint red every patch whose *pxcor* is even, and paint green all patches with odd *pxcor*. You must use the *IsEven* reporter from Q-1.

3. (5p) Create the turtle procedure *Wiggle* that will take 2 arguments: the *step-size* and *the max-angle,* and will make the turtle take one drunken step with the usual drunken technique: step in the direction it's currently heading, then turn a random amount but at most the *max-angle* in either direction.

4. (10p) (Read the whole task before starting.) Create two procedures: *Setup-Q4* and a "forever" *Go-Q4* to do the following. Create a turtle and have it make drunken steps in the *Go-Q4* procedure of either *Wiggle 0.5 20* or *Wiggle 0.2 40* with a 50% probability – in other words, each time before it moves, it flips a coin, and that way chooses which *Wiggle* to call. After 1,000 steps, it halts (unpresses the forever button).

   Imagine that every patch is a different "country". A monitor will display the value of a variable called *The-Count* which contains how many <u>totally different countries</u> the turtle has stepped in so far (i.e. visiting the same patch more than once doesn't change the count).

   You may use any globals, turtles-own or patches-own that you want.

5. (10p) Almost the same as Q4 above. Again, create *Setup-Q5* (although you may simply use *Setup*-Q4 if they're the same) and *Go-Q5*. The only difference is that now *The-Count* contains the number of times the turtle has crossed patch borders. Because the step sizes are small, the turtle will often stay on the same patch, but sometimes it will cross a border into another patch (that counts), and sometimes, though rarely, it will cross 2 borders in one step (that counts twice). This is different from Q4 because we're counting <u>border-crossings</u> -- i.e. if the turtle goes to a neighbor, that's 1 crossing, and comes back, then that's another crossing.

6. (10p) Create the reporter *AverageXcor* that takes no arguments and will report the average of the values of all of the turtles' xcors. It will report 0 if there are no turtles. You may not use the *sum* reporter built into Netlogo.

7. (5p – extra credit) Create the reporter *MaxPXY* which will be given one argument (either "x" or "y"), and will calculate the value of either the Netlogo variable *max-pxcor* or *max-pycor*. However, you may not access the built-in Netlogo variables: *max-pxcor* or *min-pxcor* or max-*pycor* or *min-pycor* or *world-width* or *world-height*.

   So, for instance, in our normal world, *MaxXY "x"* will report 16, and so will *MaxXY "y"* , but that's because the world is square. If it's not square then those numbers will not be the same.
   In other words, *MaxXY "x"* must figure out and report what the value of *max-pxcor* without help from the usual Netlogo variables. Same for *MaxXY "y"* in figuring out the value of *max-pycor*.

   The world wraps in both directions. The origin (0,0) is at the center of the world.