Survival of the Fittest
In this lab we will model evolution using a predator prey system. We will have two breeds, hawks and doves. The doves will have two sorts of behavior. In the absence of hawks they will tend to cruise along and following simple rules may tend to flock together with other doves. When they spy a hawk, they will try a variety of measures to evade the hawk. If a hawk captures a dove, another dove will give birth to a new dove in order to replace it. The genetic code of the new dove will be similar to that of its parent, but with slight mutations in each of its genes. The doves will consequently face a significant selection pressure and will evolve in response. You will write the code for this model and test certain assumptions about evolution as a result. When it comes to the testing phase I encourage you to explore the different scenarios thoroughly, in order to learn what the model predicts quantitatively about how evolution responds

Flocking
While it might be interesting to write a program that gives doves the simple instructions that lead to an emergent flocking behavior, there is in fact a preexisting model that does this quite well. Open the flocking model in models library under biology. The model has a number of sliders that constitute the simple rules that govern the behavior of the birds. Read the information tab so that you understand what these simple rules are. Examine the code to see how the rules are implemented. Spend some time doing this, as you will make this program your own. Now explore the model to get a feeling for how altering the various parameters alters the nature of the flocks that form. How do you get long lines of birds? How do you get wide flocks? Can you get a V-shaped flock? What parameter values lead to random motion of the birds? Take note of these values, for future reference.

Labwork:

Hawks and Doves

1. At the moment you have flocking turtles, but no hawks or doves. Save the program in your CAL directory and then make two new breeds: hawks and doves. Replaces all references to `turtles` in the program with references to `doves`. You will need to replace `cct` (which is short for `create-custom-turtles`) with `create-custom-doves`.
2. In the `setup` procedure create 4 hawks, set their color to red and give them random positions on the screen. In the go procedure ask the hawks to hunt, and then write a `hunt` procedure for the hawks. The `hunt` procedure will be similar to the `find-food` procedure you wrote for your amoeba. That is, let hawks have a `vision-radius` and then have them head towards the nearest dove within their vision radius (if there is one). However, to make things interesting, let the hawks have a `cruise-speed` and a `hunt-speed`. When they are not hunting (ie when they see no doves, or they have just eaten a dove) they will `wiggle` at the `cruise-speed`, but when they have spotted a dove they will head toward the dove at the `hunt-speed`. I would suggest a `cruise-speed` of 0.5 units per time step and a `hunt-speed` of 2 units per time step. As an additional change, when a hawk has caught a dove, have it take a short rest before hunting again. You could do this with a boolean variable like `just-fed?`, which you set to `false` in the `setup`, set to `true` after eating, and then set to `false` again after they have had a short cruise without hunting. Test your procedure to make sure it works. You should see your flock number diminishing to zero fairly rapidly.
3. To correct for diminishing doves, write a `birth` procedure which is called if the number of doves is ever less than the initial population. Have the right number of doves give birth to new doves to maintain the population. Make the new doves move away from their parents in a new heading.
4. Now comes the important part of the program, from the standpoint of survival of the fittest.

We need to give doves strategies for avoiding the hawks. Add an `escape-hawks` procedure for the doves and use it after the `flock` procedure in go. The `escape-hawks` procedure will be similar to the `hunt` procedure that you wrote for the hawks, except the doves will turn away form the nearest hawk within their `danger-sensing-radius`, rather than head towards it. I suggest you use the `turn-away` procedure that was written for the flocking part of the program. This procedure takes two arguments: `new-heading` and `max-turn`. For `new-heading` you would use the heading from the dove to the nearest visible hawk (`turn-away` ensures the dove turns away from this most dangerous of headings!), and `max-turn` would be an angle that represents the greatest angle they can turn in one time step you might call this angle `escape-angle` in a slider. The idea is that doves would try to turn away from the direction to the nearest hawk, but will only be able to turn a maximum of `escape-angle` in any one step.

5. As part of your `escape-hawks` procedure you will need to specify a `boost-speed,` which represents an increase above the normal speed of the doves. (Note the normal movement of doves is specified at the bottom of the flock procedure where it says fd 1. Change this line to be fd `normal-speed`). We have now added several variables governing the motion of the doves: `danger-sensing-radius`, `boost-speed`, `escape-angle`, and `normal-speed`. Add sliders for these variables in your interface. Test your changes and try to choose suitable ranges for your parameter values. Use the existing parameter values as a guide. I'd suggest a `normal-speed` around 0.5 and a `boost-speed` between 0 and 1. Note that the sum of `normal-speed` and `boost-speed` should be less than whatever you choose for the hawk's `hunt-speed`, otherwise the hawks really have little hope of catching any doves! Play around with the parameter values a bit. Try to find parameter values where the birds maintain flocking behavior while they attempt to evade the hawks. You should see interesting evasion maneuvers emerge.
6. One final thing before allowing our doves to evolve. Change the interface to 25 by 25 and set the population number to 100. Also, when you create the doves, allow the doves to have any color except red (if the color is a shade of red make them white). Now run your program. You should notice that over time, the number of different colors diminishes until finally you are left with doves of just one color. Since clearly color is not a factor in the feeding habits of your hawks, and anyway all your doves have equal fitness, why is this happening? This is actually an example of a phenomenon known as genetic drift (or random drift). When two or more species have equal fitness, but compete for resources, all but one will eventually die out. This is because if one species by chance has the larger population they will on average be more likely to reproduce and hence will be more likely to increase their population. We should be mindful of this genetic drift phenomena when we are playing the survival of the fittest game that you will write next. If selection pressure is not strong enough genetic drift can be the predominant factor causing your parameters to change.

7. Now its time to let your doves evolve. You will make some substantial changes to the interface of your model so you may want to save it as a new file (just incase!). There are a total of nine variables influencing the movement of the doves, so your genetic code will be made up of the values you assign to each of these variables. For each of the dove-related sliders in your model (except population) make a `doves-own` variable. NetLogo will complain at first because you have sliders for these variable names, but you will change the slider names soon so ignore the complaints. When you first create the doves assign each of these variables a random number (use random-float) between 0 and what you judge to be a suitable upper limit. Caution: to get realistic evolution make sure the upper limit for the two dove speeds sums to a number less than the the hawk's hunt-speed. If you don't you may create super doves by hand rather than allowing them to evolve.

8. Replace the sliders for the variables in question 7 with mutation rate sliders (eg change

`normal-speed` to `normal-speed-mutation`, choosing a range of values so that moderate, yet significant change in behavior can occur (the range of these mutation rates is best kept less than 10 percent of the range of the corresponding variables – for example, if normal speed ranges from 0 to 1 then the mutation rate should range between 0 and 0.1 .In the experimental phase of the lab you will be adjusting these parameters to see how mutation rates influence evolution.

9. In your birth procedure add a line asking newly hatched doves to mutate, and then write a mutate procedure. The mutate procedure should add to each gene (variable) a random number between –*r* and *r*, where *r* is the mutation rate for that variable. You will need to write a separate line of code for each variable in the genetic code.
10. Create monitors to keep track each variable in the genetic code. The monitors should show the average values from all the doves.
11. So that you can have a measure of the improved fitness of your doves create a new global variable called death-rate. Death-rate should measure the number of turtles killed at each time step. Now plot a graph of death-rate over time.
12. Test the model under general conditions. Allow it to run for a while – there are several changes that happen. First unfit birds are weeded out quite quickly. Then some of the gene's of the surviving birds change slowly over time. What genes change the most? Explain why? If you run the program a number of different times, with the same settings, you should be able to see species with slightly different behaviors surviving. Occasionally two species with different behaviors persist for some time together. They may have approximately equal survival rates, but for different reasons. Eventually genetic drift will eliminate one of these species. Describe the different survival strategies you observe.
13. Now lets experiment with mutation rates. Set mutation rates for all dove *speeds* to zero. Now the speeds will not change. Which genes change the most now? Does the death rate decrease? Does it go down as low as in question 12?. You should notice that some of the gene's seem to keep increasing in value, yet the death rate of the doves doesn't improve. Explain what is happening?
14. Repeat the analysis you did in each of the previous questions, but increase the number of hawks significantly. This should accelerate the selection pressure significantly. Does it alter the behavior and genetic make-up of successful doves?
15. Repeat the analysis now by setting all mutation rates high. Does this have an affect on the overall fitness? Make quantitative comparisons. How about when mutation rates are low?
16. Replace the flocking command in `go`, with a simple `wiggle` at `normal-speed` command. Return mutation rates to the values in 12. Allow the system to evolve. What do you notice about the survival rates of the doves. Is flocking a good survival strategy? Do you think your answer would be different if you changed the number of hawks to a lower number?

Survival of the Fittest
In this lab we will model evolution using a predator prey system. We will have two breeds, hawks and doves. The doves will have two sorts of behavior. In the absence of hawks they will tend to cruise along and following simple rules may tend to flock together with other doves. When they spy a hawk, they will try a variety of measures to evade the hawk. If a hawk captures a dove, another dove will give birth to a new dove in order to replace it. The genetic code of the new dove will be similar to that of its parent, but with slight mutations in each of its genes. The doves will consequently face a significant selection pressure and will evolve in response. You will write the code for this model and test certain assumptions about evolution as a result. When it comes to the testing phase I encourage you to explore the different scenarios thoroughly, in order to learn what the model predicts quantitatively about how evolution responds

Flocking
While it might be interesting to write a program that gives doves the simple instructions that lead to an emergent flocking behavior, there is in fact a preexisting model that does this quite well. Open the flocking model in models library under biology. The model has a number of sliders that constitute the simple rules that govern the behavior of the birds. Read the information tab so that you understand what these simple rules are. Examine the code to see how the rules are implemented. Spend some time doing this, as you will make this program your own. Now explore the model to get a feeling for how altering the various parameters alters the nature of the flocks that form. How do you get long lines of birds? How do you get wide flocks? Can you get a V-shaped flock? What parameter values lead to random motion of the birds? Take note of these values, for future reference.

Labwork:

Hawks and Doves
1. At the moment you have flocking turtles, but no hawks or doves. Save the program in your CAL directory and then make two new breeds: hawks and doves. Replaces all references to `turtles` in the program with references to `doves`. You will need to replace `cct` (which is short for `create-custom-turtles`) with `create-custom-doves`.
2. In the `setup` procedure create 4 hawks, set their color to red and give them random positions on the screen. In the go procedure ask the hawks to hunt, and then write a `hunt` procedure for the hawks. The `hunt` procedure will be similar to the `find-food` procedure you wrote for your amoeba. That is, let hawks have a `vision-radius` and then have them head towards the nearest dove within their vision radius (if there is one). However, to make things interesting, let the hawks have a `cruise-speed` and a `hunt-speed`. When they are not hunting (ie when they see no doves, or they have just eaten a dove) they will `wiggle` at the `cruise-speed`, but when they have spotted a dove they will head toward the dove at the `hunt-speed`. I would suggest a `cruise-speed` of 0.5 units per time step and a `hunt-speed` of 2 units per time step. As an additional change, when a hawk has caught a dove, have it take a short rest before hunting again. You could do this with a boolean variable like `just-fed?`, which you set to `false` in the `setup`, set to `true` after eating, and then set to `false` again after they have had a short cruise without hunting. Test your procedure to make sure it works. You should see your flock number diminishing to zero fairly rapidly.
3. To correct for diminishing doves, write a `birth` procedure which is called if the number of doves is ever less than the initial population. Have the right number of doves give birth to new doves to maintain the population. Make the new doves move away from their parents in a new heading.
4. Now comes the important part of the program, from the standpoint of survival of the fittest.

We need to give doves strategies for avoiding the hawks. Add an `escape-hawks` procedure for the doves and use it after the `flock` procedure in go. The `escape-hawks` procedure will be similar to the `hunt` procedure that you wrote for the hawks, except the doves will turn away form the nearest hawk within their `danger-sensing-radius`, rather than head towards it. I suggest you use the `turn-away` procedure that was written for the flocking part of the program. This procedure takes two arguments: `new-heading` and `max-turn`. For `new-heading` you would use the heading from the dove to the nearest visible hawk (`turn-away` ensures the dove turns away from this most dangerous of headings!), and `max-turn` would be an angle that represents the greatest angle they can turn in one time step you might call this angle `escape-angle` in a slider. The idea is that doves would try to turn away from the direction to the nearest hawk, but will only be able to turn a maximum of `escape-angle` in any one step.

5. As part of your `escape-hawks` procedure you will need to specify a `boost-speed,` which represents an increase above the normal speed of the doves. (Note the normal movement of doves is specified at the bottom of the flock procedure where it says fd 1. Change this line to be fd `normal-speed`). We have now added several variables governing the motion of the doves: `danger-sensing-radius`, `boost-speed`, `escape-angle`, and `normal-speed`. Add sliders for these variables in your interface. Test your changes and try to choose suitable ranges for your parameter values. Use the existing parameter values as a guide. I'd suggest a `normal-speed` around 0.5 and a `boost-speed` between 0 and 1. Note that the sum of `normal-speed` and `boost-speed` should be less than whatever you choose for the hawk's `hunt-speed`, otherwise the hawks really have little hope of catching any doves! Play around with the parameter values a bit. Try to find parameter values where the birds maintain flocking behavior while they attempt to evade the hawks. You should see interesting evasion maneuvers emerge.
6. One final thing before allowing our doves to evolve. Change the interface to 25 by 25 and set the population number to 100. Also, when you create the doves, allow the doves to have any color except red (if the color is a shade of red make them white). Now run your program. You should notice that over time, the number of different colors diminishes until finally you are left with doves of just one color. Since clearly color is not a factor in the feeding habits of your hawks, and anyway all your doves have equal fitness, why is this happening? This is actually an example of a phenomenon known as genetic drift (or random drift). When two or more species have equal fitness, but compete for resources, all but one will eventually die out. This is because if one species by chance has the larger population they will on average be more likely to reproduce and hence will be more likely to increase their population. We should be mindful of this genetic drift phenomena when we are playing the survival of the fittest game that you will write next. If selection pressure is not strong enough genetic drift can be the predominant factor causing your parameters to change.

7. Now its time to let your doves evolve. You will make some substantial changes to the interface of your model so you may want to save it as a new file (just incase!). There are a total of nine variables influencing the movement of the doves, so your genetic code will be made up of the values you assign to each of these variables. For each of the dove-related sliders in your model (except population) make a `doves-own` variable. NetLogo will complain at first because you have sliders for these variable names, but you will change the slider names soon so ignore the complaints. When you first create the doves assign each of these variables a random number (use random-float) between 0 and what you judge to be a suitable upper limit. Caution: to get realistic evolution make sure the upper limit for the two dove speeds sums to a number less than the the hawk's hunt-speed. If you don't you may create super doves by hand rather than allowing them to evolve.

8. Replace the sliders for the variables in question 7 with mutation rate sliders (eg change

`normal-speed` to `normal-speed-mutation,` choosing a range of values so that moderate, yet significant change in behavior can occur (the range of these mutation rates is best kept less than 10 percent of the range of the corresponding variables – for example, if normal speed ranges from 0 to 1 then the mutation rate should range between 0 and 0.1 .In the experimental phase of the lab you will be adjusting these parameters to see how mutation rates influence evolution.

9. In your birth procedure add a line asking newly hatched doves to mutate, and then write a mutate procedure. The mutate procedure should add to each gene (variable) a random number between –*r* and *r*, where *r* is the mutation rate for that variable. You will need to write a separate line of code for each variable in the genetic code.
10. Create monitors to keep track each variable in the genetic code. The monitors should show the average values from all the doves.
11. So that you can have a measure of the improved fitness of your doves create a new global variable called death-rate. Death-rate should measure the number of turtles killed at each time step. Now plot a graph of death-rate over time.
12. Test the model under general conditions. Allow it to run for a while – there are several changes that happen. First unfit birds are weeded out quite quickly. Then some of the gene's of the surviving birds change slowly over time. What genes change the most? Explain why? If you run the program a number of different times, with the same settings, you should be able to see species with slightly different behaviors surviving. Occasionally two species with different behaviors persist for some time together. They may have approximately equal survival rates, but for different reasons. Eventually genetic drift will eliminate one of these species. Describe the different survival strategies you observe.
13. Now lets experiment with mutation rates. Set mutation rates for all dove *speeds* to zero. Now the speeds will not change. Which genes change the most now? Does the death rate decrease? Does it go down as low as in question 12?. You should notice that some of the gene's seem to keep increasing in value, yet the death rate of the doves doesn't improve. Explain what is happening?
14. Repeat the analysis you did in each of the previous questions, but increase the number of hawks significantly. This should accelerate the selection pressure significantly. Does it alter the behavior and genetic make-up of successful doves?
15. Repeat the analysis now by setting all mutation rates high. Does this have an affect on the overall fitness? Make quantitative comparisons. How about when mutation rates are low?
16. Replace the flocking command in `go`, with a simple `wiggle` at `normal-speed` command. Return mutation rates to the values in 12. Allow the system to evolve. What do you notice about the survival rates of the doves. Is flocking a good survival strategy? Do you think your answer would be different if you changed the number of hawks to a lower number?

Survival of the Fittest

In this lab we will model evolution using a predator prey system. We will have two breeds, hawks and doves. The doves will have two sorts of behavior. In the absence of hawks they will tend to cruise along and following simple rules may tend to flock together with other doves. When they spy a hawk, they will try a variety of measures to evade the hawk. If a hawk captures a dove, another dove will give birth to a new dove in order to replace it. The genetic code of the new dove will be similar to that of its parent, but with slight mutations in each of its genes. The doves will consequently face a significant selection pressure and will evolve in response. You will write the code for this model and test certain assumptions about evolution as a result. When it comes to the testing phase I encourage you to explore the different scenarios thoroughly, in order to learn what the model predicts quantitatively about how evolution responds

Flocking

While it might be interesting to write a program that gives doves the simple instructions that lead to an emergent flocking behavior, there is in fact a preexisting model that does this quite well. Open the flocking model in models library under biology. The model has a number of sliders that constitute the simple rules that govern the behavior of the birds. Read the information tab so that you understand what these simple rules are. Examine the code to see how the rules are implemented. Spend some time doing this, as you will make this program your own. Now explore the model to get a feeling for how altering the various parameters alters the nature of the flocks that form. How do you get long lines of birds? How do you get wide flocks? Can you get a V-shaped flock? What parameter values lead to random motion of the birds? Take note of these values, for future reference.

Labwork:

Hawks and Doves

1. At the moment you have flocking turtles, but no hawks or doves. Save the program in your CAL directory and then make two new breeds: hawks and doves. Replaces all references to `turtles` in the program with references to `doves`. You will need to replace `cct` (which is short for `create-custom-turtles`) with `create-custom-doves`.
2. In the `setup` procedure create 4 hawks, set their color to red and give them random positions on the screen. In the go procedure ask the hawks to hunt, and then write a `hunt` procedure for the hawks. The `hunt` procedure will be similar to the `find-food` procedure you wrote for your amoeba. That is, let hawks have a `vision-radius` and then have them head towards the nearest dove within their vision radius (if there is one). However, to make things interesting, let the hawks have a `cruise-speed` and a `hunt-speed`. When they are not hunting (ie when they see no doves, or they have just eaten a dove) they will `wiggle` at the `cruise-speed`, but when they have spotted a dove they will head toward the dove at the `hunt-speed`. I would suggest a `cruise-speed` of 0.5 units per time step and a `hunt-speed` of 2 units per time step. As an additional change, when a hawk has caught a dove, have it take a short rest before hunting again. You could do this with a boolean variable like `just-fed?`, which you set to `false` in the `setup`, set to `true` after eating, and then set to `false` again after they have had a short cruise without hunting. Test your procedure to make sure it works. You should see your flock number diminishing to zero fairly rapidly.
3. To correct for diminishing doves, write a `birth` procedure which is called if the number of doves is ever less than the initial population. Have the right number of doves give birth to new doves to maintain the population. Make the new doves move away from their parents in a new heading.
4. Now comes the important part of the program, from the standpoint of survival of the fittest.

We need to give doves strategies for avoiding the hawks. Add an `escape-hawks` procedure for the doves and use it after the `flock` procedure in go. The `escape-hawks` procedure will be similar to the `hunt` procedure that you wrote for the hawks, except the doves will turn away form the nearest hawk within their `danger-sensing-radius`, rather than head towards it. I suggest you use the `turn-away` procedure that was written for the flocking part of the program. This procedure takes two arguments: `new-heading` and `max-turn`. For `new-heading` you would use the heading from the dove to the nearest visible hawk (`turn-away` ensures the dove turns away from this most dangerous of headings!), and `max-turn` would be an angle that represents the greatest angle they can turn in one time step you might call this angle `escape-angle` in a slider. The idea is that doves would try to turn away from the direction to the nearest hawk, but will only be able to turn a maximum of `escape-angle` in any one step.

5. As part of your `escape-hawks` procedure you will need to specify a `boost-speed,` which represents an increase above the normal speed of the doves. (Note the normal movement of doves is specified at the bottom of the flock procedure where it says fd 1. Change this line to be fd `normal-speed`). We have now added several variables governing the motion of the doves: `danger-sensing-radius`, `boost-speed`, `escape-angle`, and `normal-speed`. Add sliders for these variables in your interface. Test your changes and try to choose suitable ranges for your parameter values. Use the existing parameter values as a guide. I'd suggest a `normal-speed` around 0.5 and a `boost-speed` between 0 and 1. Note that the sum of `normal-speed` and `boost-speed` should be less than whatever you choose for the hawk's `hunt-speed`, otherwise the hawks really have little hope of catching any doves! Play around with the parameter values a bit. Try to find parameter values where the birds maintain flocking behavior while they attempt to evade the hawks. You should see interesting evasion maneuvers emerge.
6. One final thing before allowing our doves to evolve. Change the interface to 25 by 25 and set the population number to 100. Also, when you create the doves, allow the doves to have any color except red (if the color is a shade of red make them white). Now run your program. You should notice that over time, the number of different colors diminishes until finally you are left with doves of just one color. Since clearly color is not a factor in the feeding habits of your hawks, and anyway all your doves have equal fitness, why is this happening? This is actually an example of a phenomenon known as genetic drift (or random drift). When two or more species have equal fitness, but compete for resources, all but one will eventually die out. This is because if one species by chance has the larger population they will on average be more likely to reproduce and hence will be more likely to increase their population. We should be mindful of this genetic drift phenomena when we are playing the survival of the fittest game that you will write next. If selection pressure is not strong enough genetic drift can be the predominant factor causing your parameters to change.

7. Now its time to let your doves evolve. You will make some substantial changes to the interface of your model so you may want to save it as a new file (just incase!). There are a total of nine variables influencing the movement of the doves, so your genetic code will be made up of the values you assign to each of these variables. For each of the dove-related sliders in your model (except population) make a `doves-own` variable. NetLogo will complain at first because you have sliders for these variable names, but you will change the slider names soon so ignore the complaints. When you first create the doves assign each of these variables a random number (use random-float) between 0 and what you judge to be a suitable upper limit. Caution: to get realistic evolution make sure the upper limit for the two dove speeds sums to a number less than the the hawk's hunt-speed. If you don't you may create super doves by hand rather than allowing them to evolve.

8. Replace the sliders for the variables in question 7 with mutation rate sliders (eg change

`normal-speed` to `normal-speed-mutation`, choosing a range of values so that moderate, yet significant change in behavior can occur (the range of these mutation rates is best kept less than 10 percent of the range of the corresponding variables – for example, if normal speed ranges from 0 to 1 then the mutation rate should range between 0 and 0.1 .In the experimental phase of the lab you will be adjusting these parameters to see how mutation rates influence evolution.

9. In your birth procedure add a line asking newly hatched doves to mutate, and then write a mutate procedure. The mutate procedure should add to each gene (variable) a random number between –*r* and *r*, where *r* is the mutation rate for that variable. You will need to write a separate line of code for each variable in the genetic code.
10. Create monitors to keep track each variable in the genetic code. The monitors should show the average values from all the doves.
11. So that you can have a measure of the improved fitness of your doves create a new global variable called death-rate. Death-rate should measure the number of turtles killed at each time step. Now plot a graph of death-rate over time.
12. Test the model under general conditions. Allow it to run for a while – there are several changes that happen. First unfit birds are weeded out quite quickly. Then some of the gene's of the surviving birds change slowly over time. What genes change the most? Explain why? If you run the program a number of different times, with the same settings, you should be able to see species with slightly different behaviors surviving. Occasionally two species with different behaviors persist for some time together. They may have approximately equal survival rates, but for different reasons. Eventually genetic drift will eliminate one of these species. Describe the different survival strategies you observe.
13. Now lets experiment with mutation rates. Set mutation rates for all dove *speeds* to zero. Now the speeds will not change. Which genes change the most now? Does the death rate decrease? Does it go down as low as in question 12?. You should notice that some of the gene's seem to keep increasing in value, yet the death rate of the doves doesn't improve. Explain what is happening?
14. Repeat the analysis you did in each of the previous questions, but increase the number of hawks significantly. This should accelerate the selection pressure significantly. Does it alter the behavior and genetic make-up of successful doves?
15. Repeat the analysis now by setting all mutation rates high. Does this have an affect on the overall fitness? Make quantitative comparisons. How about when mutation rates are low?
16. Replace the flocking command in `go`, with a simple `wiggle` at `normal-speed` command. Return mutation rates to the values in 12. Allow the system to evolve. What do you notice about the survival rates of the doves. Is flocking a good survival strategy? Do you think your answer would be different if you changed the number of hawks to a lower number?

Survival of the Fittest
In this lab we will model evolution using a predator prey system. We will have two breeds, hawks and doves. The doves will have two sorts of behavior. In the absence of hawks they will tend to cruise along and following simple rules may tend to flock together with other doves. When they spy a hawk, they will try a variety of measures to evade the hawk. If a hawk captures a dove, another dove will give birth to a new dove in order to replace it. The genetic code of the new dove will be similar to that of its parent, but with slight mutations in each of its genes. The doves will consequently face a significant selection pressure and will evolve in response. You will write the code for this model and test certain assumptions about evolution as a result. When it comes to the testing phase I encourage you to explore the different scenarios thoroughly, in order to learn what the model predicts quantitatively about how evolution responds

Flocking
While it might be interesting to write a program that gives doves the simple instructions that lead to an emergent flocking behavior, there is in fact a preexisting model that does this quite well. Open the flocking model in models library under biology. The model has a number of sliders that constitute the simple rules that govern the behavior of the birds. Read the information tab so that you understand what these simple rules are. Examine the code to see how the rules are implemented. Spend some time doing this, as you will make this program your own. Now explore the model to get a feeling for how altering the various parameters alters the nature of the flocks that form. How do you get long lines of birds? How do you get wide flocks? Can you get a V-shaped flock? What parameter values lead to random motion of the birds? Take note of these values, for future reference.

Labwork:

Hawks and Doves
1. At the moment you have flocking turtles, but no hawks or doves. Save the program in your CAL directory and then make two new breeds: hawks and doves. Replaces all references to `turtles` in the program with references to `doves`. You will need to replace `cct` (which is short for `create-custom-turtles`) with `create-custom-doves`.
2. In the `setup` procedure create 4 hawks, set their color to red and give them random positions on the screen. In the go procedure ask the hawks to hunt, and then write a `hunt` procedure for the hawks. The `hunt` procedure will be similar to the `find-food` procedure you wrote for your amoeba. That is, let hawks have a `vision-radius` and then have them head towards the nearest dove within their vision radius (if there is one). However, to make things interesting, let the hawks have a `cruise-speed` and a `hunt-speed`. When they are not hunting (ie when they see no doves, or they have just eaten a dove) they will `wiggle` at the `cruise-speed`, but when they have spotted a dove they will head toward the dove at the `hunt-speed`. I would suggest a `cruise-speed` of 0.5 units per time step and a `hunt-speed` of 2 units per time step. As an additional change, when a hawk has caught a dove, have it take a short rest before hunting again. You could do this with a boolean variable like `just-fed?`, which you set to `false` in the `setup`, set to `true` after eating, and then set to `false` again after they have had a short cruise without hunting. Test your procedure to make sure it works. You should see your flock number diminishing to zero fairly rapidly.
3. To correct for diminishing doves, write a `birth` procedure which is called if the number of doves is ever less than the initial population. Have the right number of doves give birth to new doves to maintain the population. Make the new doves move away from their parents in a new heading.
4. Now comes the important part of the program, from the standpoint of survival of the fittest.

We need to give doves strategies for avoiding the hawks. Add an `escape-hawks` procedure for the doves and use it after the `flock` procedure in go. The `escape-hawks` procedure will be similar to the `hunt` procedure that you wrote for the hawks, except the doves will turn away form the nearest hawk within their `danger-sensing-radius`, rather than head towards it. I suggest you use the `turn-away` procedure that was written for the flocking part of the program. This procedure takes two arguments: `new-heading` and `max-turn`. For `new-heading` you would use the heading from the dove to the nearest visible hawk (`turn-away` ensures the dove turns away from this most dangerous of headings!), and `max-turn` would be an angle that represents the greatest angle they can turn in one time step you might call this angle `escape-angle` in a slider. The idea is that doves would try to turn away from the direction to the nearest hawk, but will only be able to turn a maximum of `escape-angle` in any one step.

5. As part of your `escape-hawks` procedure you will need to specify a `boost-speed,` which represents an increase above the normal speed of the doves. (Note the normal movement of doves is specified at the bottom of the flock procedure where it says fd 1. Change this line to be fd `normal-speed`). We have now added several variables governing the motion of the doves: `danger-sensing-radius`, `boost-speed`, `escape-angle`, and `normal-speed`. Add sliders for these variables in your interface. Test your changes and try to choose suitable ranges for your parameter values. Use the existing parameter values as a guide. I'd suggest a `normal-speed` around 0.5 and a `boost-speed` between 0 and 1. Note that the sum of `normal-speed` and `boost-speed` should be less than whatever you choose for the hawk's `hunt-speed`, otherwise the hawks really have little hope of catching any doves! Play around with the parameter values a bit. Try to find parameter values where the birds maintain flocking behavior while they attempt to evade the hawks. You should see interesting evasion maneuvers emerge.
6. One final thing before allowing our doves to evolve. Change the interface to 25 by 25 and set the population number to 100. Also, when you create the doves, allow the doves to have any color except red (if the color is a shade of red make them white). Now run your program. You should notice that over time, the number of different colors diminishes until finally you are left with doves of just one color. Since clearly color is not a factor in the feeding habits of your hawks, and anyway all your doves have equal fitness, why is this happening? This is actually an example of a phenomenon known as genetic drift (or random drift). When two or more species have equal fitness, but compete for resources, all but one will eventually die out. This is because if one species by chance has the larger population they will on average be more likely to reproduce and hence will be more likely to increase their population. We should be mindful of this genetic drift phenomena when we are playing the survival of the fittest game that you will write next. If selection pressure is not strong enough genetic drift can be the predominant factor causing your parameters to change.

7. Now its time to let your doves evolve. You will make some substantial changes to the interface of your model so you may want to save it as a new file (just incase!). There are a total of nine variables influencing the movement of the doves, so your genetic code will be made up of the values you assign to each of these variables. For each of the dove-related sliders in your model (except population) make a `doves-own` variable. NetLogo will complain at first because you have sliders for these variable names, but you will change the slider names soon so ignore the complaints. When you first create the doves assign each of these variables a random number (use random-float) between 0 and what you judge to be a suitable upper limit. Caution: to get realistic evolution make sure the upper limit for the two dove speeds sums to a number less than the the hawk's hunt-speed. If you don't you may create super doves by hand rather than allowing them to evolve.

8. Replace the sliders for the variables in question 7 with mutation rate sliders (eg change

`normal-speed` to `normal-speed-mutation`, choosing a range of values so that moderate, yet significant change in behavior can occur (the range of these mutation rates is best kept less than 10 percent of the range of the corresponding variables – for example, if normal speed ranges from 0 to 1 then the mutation rate should range between 0 and 0.1 .In the experimental phase of the lab you will be adjusting these parameters to see how mutation rates influence evolution.

9. In your birth procedure add a line asking newly hatched doves to mutate, and then write a mutate procedure. The mutate procedure should add to each gene (variable) a random number between –*r* and *r*, where *r* is the mutation rate for that variable. You will need to write a separate line of code for each variable in the genetic code.
10. Create monitors to keep track each variable in the genetic code. The monitors should show the average values from all the doves.
11. So that you can have a measure of the improved fitness of your doves create a new global variable called death-rate. Death-rate should measure the number of turtles killed at each time step. Now plot a graph of death-rate over time.
12. Test the model under general conditions. Allow it to run for a while – there are several changes that happen. First unfit birds are weeded out quite quickly. Then some of the gene's of the surviving birds change slowly over time. What genes change the most? Explain why? If you run the program a number of different times, with the same settings, you should be able to see species with slightly different behaviors surviving. Occasionally two species with different behaviors persist for some time together. They may have approximately equal survival rates, but for different reasons. Eventually genetic drift will eliminate one of these species. Describe the different survival strategies you observe.
13. Now lets experiment with mutation rates. Set mutation rates for all dove *speeds* to zero. Now the speeds will not change. Which genes change the most now? Does the death rate decrease? Does it go down as low as in question 12?. You should notice that some of the gene's seem to keep increasing in value, yet the death rate of the doves doesn't improve. Explain what is happening?
14. Repeat the analysis you did in each of the previous questions, but increase the number of hawks significantly. This should accelerate the selection pressure significantly. Does it alter the behavior and genetic make-up of successful doves?
15. Repeat the analysis now by setting all mutation rates high. Does this have an affect on the overall fitness? Make quantitative comparisons. How about when mutation rates are low?
16. Replace the flocking command in `go`, with a simple `wiggle` at `normal-speed` command. Return mutation rates to the values in 12. Allow the system to evolve. What do you notice about the survival rates of the doves. Is flocking a good survival strategy? Do you think your answer would be different if you changed the number of hawks to a lower number?

Survival of the Fittest
In this lab we will model evolution using a predator prey system. We will have two breeds, hawks and doves. The doves will have two sorts of behavior. In the absence of hawks they will tend to cruise along and following simple rules may tend to flock together with other doves. When they spy a hawk, they will try a variety of measures to evade the hawk. If a hawk captures a dove, another dove will give birth to a new dove in order to replace it. The genetic code of the new dove will be similar to that of its parent, but with slight mutations in each of its genes. The doves will consequently face a significant selection pressure and will evolve in response. You will write the code for this model and test certain assumptions about evolution as a result. When it comes to the testing phase I encourage you to explore the different scenarios thoroughly, in order to learn what the model predicts quantitatively about how evolution responds

Flocking
While it might be interesting to write a program that gives doves the simple instructions that lead to an emergent flocking behavior, there is in fact a preexisting model that does this quite well. Open the flocking model in models library under biology. The model has a number of sliders that constitute the simple rules that govern the behavior of the birds. Read the information tab so that you understand what these simple rules are. Examine the code to see how the rules are implemented. Spend some time doing this, as you will make this program your own. Now explore the model to get a feeling for how altering the various parameters alters the nature of the flocks that form. How do you get long lines of birds? How do you get wide flocks? Can you get a V-shaped flock? What parameter values lead to random motion of the birds? Take note of these values, for future reference.

Labwork:

Hawks and Doves

1. At the moment you have flocking turtles, but no hawks or doves. Save the program in your CAL directory and then make two new breeds: hawks and doves. Replaces all references to `turtles` in the program with references to `doves`. You will need to replace `cct` (which is short for `create-custom-turtles`) with `create-custom-doves`.
2. In the `setup` procedure create 4 hawks, set their color to red and give them random positions on the screen. In the go procedure ask the hawks to hunt, and then write a `hunt` procedure for the hawks. The `hunt` procedure will be similar to the `find-food` procedure you wrote for your amoeba. That is, let hawks have a `vision-radius` and then have them head towards the nearest dove within their vision radius (if there is one). However, to make things interesting, let the hawks have a `cruise-speed` and a `hunt-speed`. When they are not hunting (ie when they see no doves, or they have just eaten a dove) they will `wiggle` at the `cruise-speed`, but when they have spotted a dove they will head toward the dove at the `hunt-speed`. I would suggest a `cruise-speed` of 0.5 units per time step and a `hunt-speed` of 2 units per time step. As an additional change, when a hawk has caught a dove, have it take a short rest before hunting again. You could do this with a boolean variable like `just-fed?`, which you set to `false` in the `setup`, set to `true` after eating, and then set to `false` again after they have had a short cruise without hunting. Test your procedure to make sure it works. You should see your flock number diminishing to zero fairly rapidly.
3. To correct for diminishing doves, write a `birth` procedure which is called if the number of doves is ever less than the initial population. Have the right number of doves give birth to new doves to maintain the population. Make the new doves move away from their parents in a new heading.
4. Now comes the important part of the program, from the standpoint of survival of the fittest.

We need to give doves strategies for avoiding the hawks. Add an `escape-hawks` procedure for the doves and use it after the `flock` procedure in go. The `escape-hawks` procedure will be similar to the `hunt` procedure that you wrote for the hawks, except the doves will turn away form the nearest hawk within their `danger-sensing-radius`, rather than head towards it. I suggest you use the `turn-away` procedure that was written for the flocking part of the program. This procedure takes two arguments: `new-heading` and `max-turn`. For `new-heading` you would use the heading from the dove to the nearest visible hawk (`turn-away` ensures the dove turns away from this most dangerous of headings!), and `max-turn` would be an angle that represents the greatest angle they can turn in one time step you might call this angle `escape-angle` in a slider. The idea is that doves would try to turn away from the direction to the nearest hawk, but will only be able to turn a maximum of `escape-angle` in any one step.

5. As part of your `escape-hawks` procedure you will need to specify a `boost-speed`, which represents an increase above the normal speed of the doves. (Note the normal movement of doves is specified at the bottom of the flock procedure where it says fd 1. Change this line to be fd `normal-speed`). We have now added several variables governing the motion of the doves: `danger-sensing-radius`, `boost-speed`, `escape-angle`, and `normal-speed`. Add sliders for these variables in your interface. Test your changes and try to choose suitable ranges for your parameter values. Use the existing parameter values as a guide. I'd suggest a `normal-speed` around 0.5 and a `boost-speed` between 0 and 1. Note that the sum of `normal-speed` and `boost-speed` should be less than whatever you choose for the hawk's `hunt-speed`, otherwise the hawks really have little hope of catching any doves! Play around with the parameter values a bit. Try to find parameter values where the birds maintain flocking behavior while they attempt to evade the hawks. You should see interesting evasion maneuvers emerge.
6. One final thing before allowing our doves to evolve. Change the interface to 25 by 25 and set the population number to 100. Also, when you create the doves, allow the doves to have any color except red (if the color is a shade of red make them white). Now run your program. You should notice that over time, the number of different colors diminishes until finally you are left with doves of just one color. Since clearly color is not a factor in the feeding habits of your hawks, and anyway all your doves have equal fitness, why is this happening? This is actually an example of a phenomenon known as genetic drift (or random drift). When two or more species have equal fitness, but compete for resources, all but one will eventually die out. This is because if one species by chance has the larger population they will on average be more likely to reproduce and hence will be more likely to increase their population. We should be mindful of this genetic drift phenomena when we are playing the survival of the fittest game that you will write next. If selection pressure is not strong enough genetic drift can be the predominant factor causing your parameters to change.

7. Now its time to let your doves evolve. You will make some substantial changes to the interface of your model so you may want to save it as a new file (just incase!). There are a total of nine variables influencing the movement of the doves, so your genetic code will be made up of the values you assign to each of these variables. For each of the dove-related sliders in your model (except population) make a `doves-own` variable. NetLogo will complain at first because you have sliders for these variable names, but you will change the slider names soon so ignore the complaints. When you first create the doves assign each of these variables a random number (use random-float) between 0 and what you judge to be a suitable upper limit. Caution: to get realistic evolution make sure the upper limit for the two dove speeds sums to a number less than the the hawk's hunt-speed. If you don't you may create super doves by hand rather than allowing them to evolve.

8. Replace the sliders for the variables in question 7 with mutation rate sliders (eg change

`normal-speed` to `normal-speed-mutation`, choosing a range of values so that moderate, yet significant change in behavior can occur (the range of these mutation rates is best kept less than 10 percent of the range of the corresponding variables – for example, if normal speed ranges from 0 to 1 then the mutation rate should range between 0 and 0.1 .In the experimental phase of the lab you will be adjusting these parameters to see how mutation rates influence evolution.

9. In your birth procedure add a line asking newly hatched doves to mutate, and then write a mutate procedure. The mutate procedure should add to each gene (variable) a random number between –*r* and *r*, where *r* is the mutation rate for that variable. You will need to write a separate line of code for each variable in the genetic code.
10. Create monitors to keep track each variable in the genetic code. The monitors should show the average values from all the doves.
11. So that you can have a measure of the improved fitness of your doves create a new global variable called death-rate. Death-rate should measure the number of turtles killed at each time step. Now plot a graph of death-rate over time.
12. Test the model under general conditions. Allow it to run for a while – there are several changes that happen. First unfit birds are weeded out quite quickly. Then some of the gene's of the surviving birds change slowly over time. What genes change the most? Explain why? If you run the program a number of different times, with the same settings, you should be able to see species with slightly different behaviors surviving. Occasionally two species with different behaviors persist for some time together. They may have approximately equal survival rates, but for different reasons. Eventually genetic drift will eliminate one of these species. Describe the different survival strategies you observe.
13. Now lets experiment with mutation rates. Set mutation rates for all dove *speeds* to zero. Now the speeds will not change. Which genes change the most now? Does the death rate decrease? Does it go down as low as in question 12?. You should notice that some of the gene's seem to keep increasing in value, yet the death rate of the doves doesn't improve. Explain what is happening?
14. Repeat the analysis you did in each of the previous questions, but increase the number of hawks significantly. This should accelerate the selection pressure significantly. Does it alter the behavior and genetic make-up of successful doves?
15. Repeat the analysis now by setting all mutation rates high. Does this have an affect on the overall fitness? Make quantitative comparisons. How about when mutation rates are low?
16. Replace the flocking command in `go`, with a simple `wiggle` at `normal-speed` command. Return mutation rates to the values in 12. Allow the system to evolve. What do you notice about the survival rates of the doves. Is flocking a good survival strategy? Do you think your answer would be different if you changed the number of hawks to a lower number?