UNIT 20. END OF THE COURSE

Welcome to the end of the course!

What did you learn in this course? You may think that you learned Processing, but actually, you learned fundamental programming techniques. If you stick with computers, regardless of the language you us you will most likely see integers, floating points, functions, booleans and ifs, loops and arrays, in your next language. More importantly, the thought processes you have been applying to use these tools to solve logic transcend computer science.

For example, here is a program in a language that you have most likely never seen before. This is a real program, can you guess what it does? Can you quickly figure out what some of the constructs are?

```
var
  endPoint, startPoint, midPoint, target,
  indexFound : Integer;
  data : Array[0..11] of Integer
       = (1,1,1,1,2,4,5,5,10,90,100,1000);
Begin
 startPoint := 0;
 endPoint := Length(data)-1;
 target := 101;
 indexFound := -1;
 while (startPoint <= endPoint) AND (indexFound = -1) do
 begin
   midPoint := Floor((startPoint + endPoint) /2);
   if data[midPoint] = target then
     indexFound := midpoint
   else if data[midPoint] < target then
     startPoint := midPoint+1
   else
     endPoint := midPoint -1;
 end;
 writeln(indexFound);
End.
```

This is a program written in Pascal, a language used for teaching. Things are a little different, but should be reasonably comfortable. See? You learned fundamental programming.

Because of this, as you advance, many text books will not give examples in any

particular language. One common method is to use *pseudo-code*, readable descriptions of algorithms that don't match any particular language. For example, here is some pseudocode you may see in a text book:

```
for j ←1 to length(A)-1
      key ← A[ j ]
      i ← j - 1
      while i >= 0 and A [ i ] > key
                A[ i +1 ] ← A[ i ]
                i ← i -1
      A [i +1] ← key
```

You may not understand exactly the nuances of what the outcome is, until you work through it, but the step by step pieces should make reasonable sense.

You are done with the course! As a bonus, this book contains two extra chapters that are not part of the course itself. As such, these chapters do not have the exercises or formatting of the previous chapters. First, you can get a quick introduction to Object Oriented Programming, and second, you can see how to move from Processing to raw Java.

You're done! Any feedback, suggestions, corrections, etc., are welcome: young@ca.umanitoba.ca

