Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Class: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Session 2: Points, Lines, Parallels and Perpendiculars**

\*All exercises should be opened in the editor and ran in the terminal as stated in the introduction.

**Exercise:** *Open* p01points.hs

*Run the program and notice that the messages show 0 instead of the actual distances between points. Fix the program, so that the message shows the right distances.*

*Use the distance formula and the function* sqrt *to compute the distance between points. The expression* sqrt a *returns the square root of a number* a. *Notice that there are no parentheses or commas between* sqrt *and* a, *but just a space.*

*After you fix the program, run it 5 times and check that it always produces a correct result.*

**Exercise:** *Open* p02collinear.hs

*Run the program and notice that the message shows X where it should show either A, B or C.*

*Look at lines 13-16 in the program. They handle the different possible scenarios that can occur when you get 3 random points. When you run the program, it looks at lines starting with a vertical bar and chooses the first one that holds true at that time. These statements are called* conditional statements.

*Replace the X in each line with either A, B or C, so that the conditional statements are true.*

*The symbol =~ means approximately equal to. It is necessary because there is always some rounding error in computations, and mathematical equalities often do not hold exactly.*

*Run the program 5 times and check that it always produces a correct result.*

**Exercise:** *Open* p03triangle.hs

*Run the program. The point C’ is at the intersection of the line AB with a perpendicular line passing through C. The height of the triangle is the length of the segment CC’.*

*You can use the function* dist *to compute the distance between two points. For example, the distance between C and C’ can be expressed as:* dist c c’

*That is, you write the word* dist*, then a space, then the name the first point, then another space, and finally the name of the second point.*

*Use the function* dist *with the right parameters to calculate the perimeter and the area of a random triangle. Save your file.*

*Run the program 5 times and check that it always produces a correct result.*

**Exercise:** *Open* p04fourpoints.hs

*Find the area and the perimeter of a random quadrilateral. Consider the quadrilateral as the union of 2 triangles that share a diagonal. Compute the heights of those triangles using the functions* dist *and* projection.

*The expression* projection (a,b) c *calculates the point at the intersection of a line through points* a *and* b *with a perpendicular line passing through point* c.

*Run the program 5 times and check that it always produces a correct result.*

**Exercise:** *Open* p04ushape.hs

*Your task is to write a program that calculates the area and the perimeter of this U-shape. Run the program several times to see how the shape changes.*

*There are many ways to go about this calculation, but all of them involve breaking this shape into smaller, simpler shapes and then add the areas of those components.*

*After you write the program, run it 5 times to check that it always produces a correct result.*

**Exercise:** *Open* p05parallel.hs

*There is a conditional statement in this program that is wrong. The condition to check for parallel lines should not be whether the* angle a b c *is 90 degrees or not, but another one also dealing with angles. Replace the wrong condition with the correct one.*

*The expression* angle a b c *is used to calculate the measure of an angle with vertex at point* b *and sides containing points* a *and* c, *respectively. The angle measure is given in degrees.*

*After you fix the code, you may need to run this program 15 or 20 times before you get a feeling of whether it is correct or incorrect.*

**Exercise:** *Open* p06perpendicular.hs

*Modify this program so that a random rectangle is shown on the screen. The rectangle may or may not include any of the points A,B,C or D, depending on how you go about solving this problem.*

*There are many possible ways to solve this problem, as the ultimate goal is just to show a random rectangle on the screen using the functions you have been practicing. Two additional functions are provided:* perpendicular *and* line\_line.

*The expression* perpendicular (a,b) c *returns the line that is perpendicular to line AB and passes through point C.*

*The expression* line\_line l1 l2 *returns a list with either 0 or 1 points at the intersection of lines* l1 *and* l2.

*Note that in this system, a line is always denoted by a pair of points.*

**Exercise:** *Open* p07rhombus.hs

*Modify the program so that a random rhombus is shown on the screen. You may need to copy code that you wrote for* p06perpendicular.hs *into this file. You may also use the function* midpoint. *All the functions you may need are listed in the* import Unit02 *statement on line 3.*

*The expression* midpoint a b *returns the midpoint of the segment AB.*

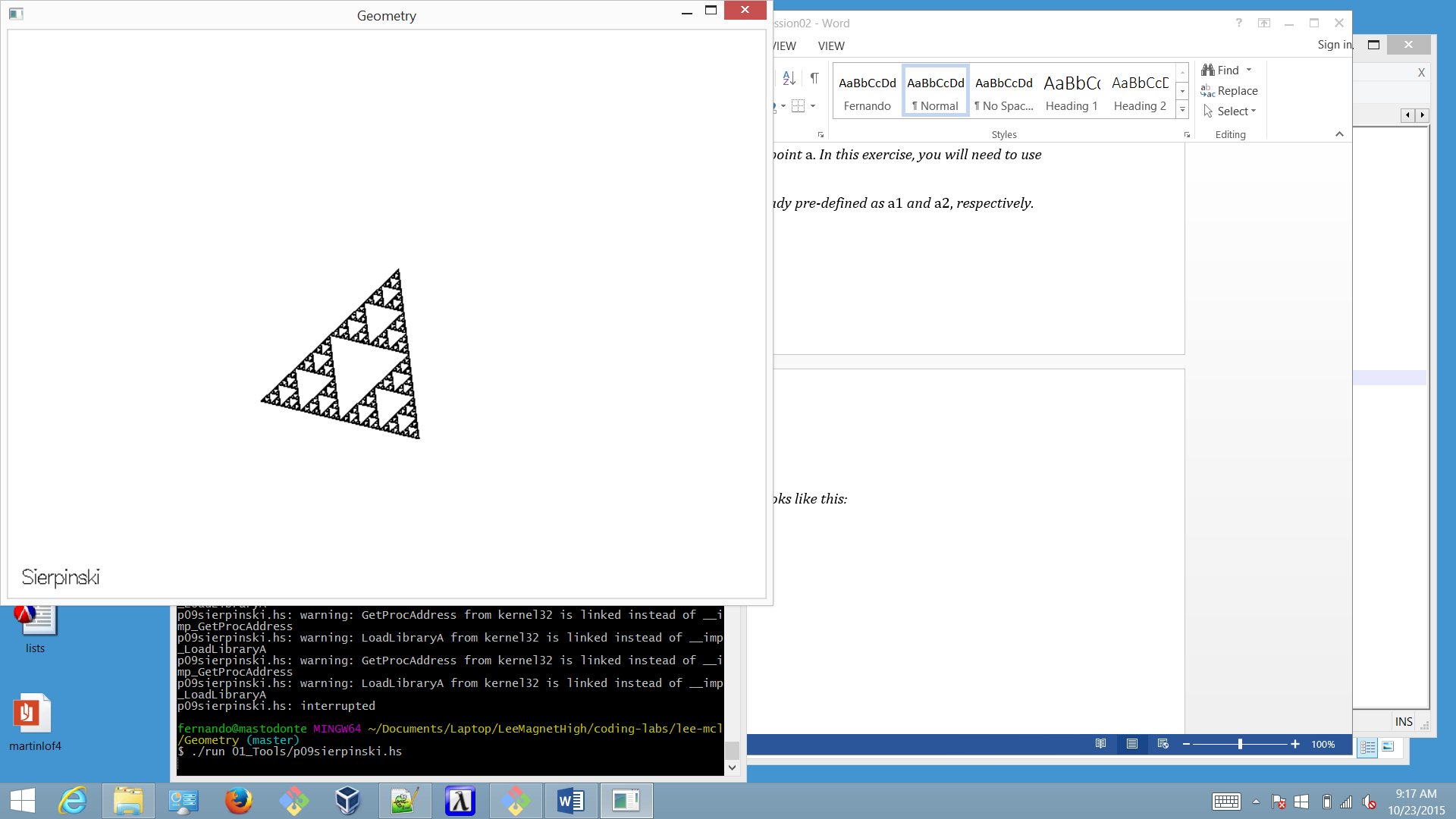
**Exercise:**  *Open* p08square.hs

*Draw a random square with one vertex at point* a. *In this exercise, you will need to use coordinates to calculate the other vertices.*

*The* x *and* y *coordinates of point* a *are already pre-defined as* a1 *and* a2, *respectively.*

**Exercise:** *Open* p09sierpinski.hs

*Write code to draw a random Sierpinski Triangle that looks like this:*



*Insert your code where it says “Your code should go here”.*