CSI 31 / Homework for Weeks 4 & 5

The first four exercises are on pages 80 – 82 in 3rd ed.(76 – 80 in 2nd ed.)

Remember—every program must tell the user what the program does (Introduction).

1. (6) Two points in a plane (*x*-*y* coordinate system) are specified using the coordinates (*x*1, *y*1) and (*x*2, *y*2). Write a program that gets the points from the user and

(a) calculates the slope of a line through two (non-vertical) points;

(c) outputs the slope to the screen.

(d) If the user enters two points on a vertical line, the denominator of the slope formula will be zero. Fix the program so that the program won’t fail if the points are on a vertical line.

(e) (Optional) Write an equation for the line determined by the two points and output it to the screen.

2. (7) Two points in a plane (*x*-*y* coordinate system) are specified using the coordinates (*x*1, *y*1) and (*x*2, *y*2). Write a program that gets the points from the user and

(a) calculates the distance between them;

(b) outputs length of the line segment to the screen.

3. (9) Write a program to calculate the area of a triangle given the lengths of its three sides a, b, c, using Heron’s formula:

A = sqrt( s(s – a)(s – b)(s – c)) where s = (a + b + c)/2, the semi-perimeter

Get the lengths of the sides from the user. Calculate the area of the triangle and output the area to the screen. Under certain circumstances, the lengths of three line segments do not form a triangle (i. e., the sum of any two sides must be greater than the third – a + b > c, a + c > b and b + c > a). Revise the program to check that the line segments will form a triangle and if not, inform the user and ask for different lengths.

4. (17) Write an algorithm for computing square roots of positive numbers. One way is to use a guess and check approach (i, e., guess the square root of 17 is 4.1, check by squaring 4.1

[(4.1)(4.1) = 16.81], so 4.1 is too small; try 4.12 [(4.12)(4.12) = 16.9744] etc.).

A programmable way of making guesses is to use Newton’s method. Suppose *x* is the number we want the square root of and *guess* is the current guessed answer. The guess can be improved upon by using *guess =*  (*guess* + (*x/guess*)) / 2 as the next guess.

Write the **design specifications** and the **program** that implements Newton’s method.

Get the number to take the square root of (*x*) and the number of times to improve the guess from the user. Starting with an initial *guess* = *x* / 2, your program should loop the specified number of times applying Newton’s method and report the final value of *guess*. You should also calculate the absolute value of difference (*difference*) between the square of your final *guess [*gues\*\*2 and the value *x* to determine how close *guess* is to the square root of *x*.

Output both *guess* and *difference* rounded to two decimal places to the screen with appropriate messages such as:

“The square root of”, *x*, “ís approximately”, round( *guess,*2)

“The difference between”, *x*, “and the square of”, *guess*, “is”, round(*difference*,2)

5. (#7, page 118) Circle intersection. Write a program that computes the intersection of a circle with a horizontal line and displays the information textually and graphically.

**Inpu**t: Radius of the circle *r* and the *y*-intercept of the line.

**Output**: Draw a circle centered at (0,0) the given radius in a window with coordinates running from -10, -10 to 10,10. Draw a horizontal line across the window with the given *y*-intercept. Draw the two points of intersection in red. Print out the *x* values of the points of intersection.

**Formula**: 