CS 5696 Functional Design & Programming
Spring Semester, 2015
Assignment 1

San Diego State University -- This page last updated 8/30/15

Due Sept 9 11:59 PM
Version 1.1

You can use any static method in the java.lang.Math class in this assignment. Each problem is worth 10 points. In problem 1-5 make sure that your function names and arguments are as given in the problem description as unit tests will be used to validate your answers. Each question asks you to write a function. You can write more that one function if you find it useful.

1. We can use a map to describe an item on a restaurant bill. For example {:name “Green Tea Ice Cream” :price 2.5 :quantity 2}. We can represent the bill as a vector of maps. Write a Clojure function **bill-total** whose one argument is a vector of such maps and returns the total of the bill. For example

```
(def bill [{:name “Green Tea Ice Cream” :price 2.5 :quantity 2}
           {:price 1.0 :name “Sticky Rice” :quantity 1}])
(bill-total bill) returns 6.0
```

2. Often people will order additional times. Write a function **add-to-bill** that accepts two arguments. The first is a bill as above. The second argument is a vector of additional items. The method returns a new bill with the additional items. For example:

```
(def items [{:price 2.1 :name “Mango” :quantity 1} { :quantity 1:price 1.0 :name “Sticky
Rice” }

(add-to-bill bill items) returns [{:name “Green Tea Ice Cream” :price 2.5 :quantity 2}
           {:price 1.0 :name “Sticky Rice” :quantity 2}
           {:price 2.1 :name “Mango” :quantity 1}]
```

We can represent a polynomial of one variable as a vector of vectors. For example $x^2 + 3*x - 1$ can be represented by [[1 2] [3 1] [-1 0]]. Here the first element of the inner vector is the coefficient and the second element is the exponent of variable.

3. Write a Clojure function, **make-poly**, with one argument, a polynomial in the form given above. The function returns a function, call it polynomial. This polynomial function accepts one argument, a number, and returns the value of the polynomial evaluated on the input. So we will get

```
(def example (poly-maker [[1 2] [3 1] [-1 0]]))
(example 2) will return 9.0
(map example [0 1 2 3 4 5]) will return (-1.0 3.0 9.0 17.0 27.0 39.0)
```

4. Write a function **differentiate** that has one argument, a polynomial in the above format. The function returns the derivative of the polynomial. So

```
(differentiate [[1 2] [3 1] [-1 0]]) returns [[2 1] [3 0]]
(differentiate [[3 4] [5 2] [6 1]]) returns [[12 3] [10 1] [6 0]]
```

5. Given a polynomial, call it p(x), we want to find a value of x where p(x) = 0. That is x is a root of the polynomial. Let p'(x) be the derivative of p(x). Select a value x_0 and let $x_1 = x_0 - p(x_0)/p'(x_0)$, $x_2 = x_1 - p(x_1)/p'(x_1)$, ... , $x_n - p(x_{n-1})/p'(x_{n-1})$. In most cases $x_0, x_1, x_2, x_3, \dots x_n$ converges to a root of the polynomial. So to find a root of the polynomial compute $x_0, x_1, x_2, x_3, \dots x_n$ until $| x_n - x_{n-1} |$ is small. Then x_n is a good approximate of a root of the polynomial. Write a Clojure function **find-root** with three arguments. The first is float that is how small we want $| x_n - x_{n-1} |$ to be. The second is polynomial vector that we want to find th e root of. The third argument is a guess for x_0. The function find-root return x_n. For example

```
(def poly1 [[1 2] [2 1] [1 0]])          ;(x+1)(x+1) so root = -1
(def poly2 [[1 2]  [-1 0]])              ;(x+1)(x-1)  so roots are 1, -1
(def poly3 [[6 2] [1 1] [-1 0]])         ;(2x+1)(3x-1) so roots are -1/2 and 1/3

(find-root 0.0001 poly1 10)   returns -0.999832
(find-root 0.0001 poly2 10)   returns 1.000005
(find-root 0.0001 poly2 -10)   returns -1.000005
(find-root 0.0001 poly3 10)   returns 0.3333333
```

6. A common example in Object-Oriented books is a Bank Account Class. Write Clojure function(s) that allow you to deposit or withdrawal money from a bank account.

What to Turn in

Answer all questions in a single Clojure file (file extension .clj). Use a comment to separate and label each questions. Place the questions in order in your file. Zip up the file and turn in your zipped file using assignment 1 link on the course portal.

Late Penalty

An assignment turned in 1-7 days late, will lose 5% of the total value of the assignment per day late. The eight day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.

Document History

8/30/2015 Version 1.1

Corrected typo in problem 5. Replaced $x_1 = x_0 - p(x_0)/p(x_0)$, $x_2 = x_1 - p(x_1)/p(x_1)$, ... , $x_n - p(x_{n-1})/p(x_{n-1})$ with $x_1 = x_0 - p(x_0)/p'(x_0)$, $x_2 = x_1 - p(x_1)/p'(x_1)$, ... , $x_n - p(x_{n-1})/p'(x_{n-1})$