CSC 402/502 Homework 4 Instructor: Jeff Ward
Covers: F#: Collections (Sections 5.1-5.3) 30 pts

For this assignment create a file called HW4.fs. All of your code for this assignment should be placed in this .fs file.

If you are using JetBrains Rider as your IDE then you can create the file and runnable project as follows: Hit File->New... . Select "Console Application" under .NET Core. Give the solution a name (e.g. "HW4" without the quotes). The dialog box will fill in the same name (e.g. "HW4") as the name of the project. Click "Create". Rider will create a file named Program.fs under the HW4 project.

Your work for this assignment will be submitted through Canvas, but automatically graded by a program called Mimir. In order to prepare your code for this, you need to perform the following three simple steps to your Program.fs file.

IMPORTANT: Right-click on Program.fs file , select Edit->Rename..., and change the file's name to HW4.fs.

IMPORTANT: Put the declaration **`module HW4`** at the top of HW4.fs.

IMPORTANT: At the bottom of HW4.fs file delete the `[<EntryPoint>]` directive and all of the code that constitutes the main function. Mimir is going to use the entry point and main function from a file called HW4Test.fs.

I've posted a copy of HW4Test.fs on Canvas, with the calls to your functions commented out. So you have two options for how you test your code as you work the problems: (1) use the REPL as in the earlier assignments, or (2) uncomment the tests in HW4Test.fs and hit Run->Run HW4. Use whichever approach you find most convenient. In order to include HW4Test.fs in your project, first download the file to your computer. Then drag it into your HW project. Make sure that HW4Test.fs is listed *below* HW4.fs in the project: Unlike the Java and C# languages, the F# language requires that the files that define functions are listed in a project before the files that use those functions.

There are ten problems in this assignment, each of which asks you to write a single function. Although some of the concepts (such as *folds*) take getting use to, none of these functions need to be very long. In fact, the instructor's solutions for each of the first nine problems are only one line of code each. You will probably get more out of this assignment if you strive to be similarly concise. For example, none of the first nine problems require if-expressions, and none of the problems at all require recursion or pattern matching.

Problem 1 (Lists – 3 points) Use the List.fold function to define a function *listProduct* that takes a list of integers and returns their product.

```
val listProduct : list:int list -> int
```

```
listProduct [2;3;5;7];;
val it : int = 210
```

```
listProduct [];;
val it : int = 1
```

Problem 2 (Lists – 1 point) In my copy of the textbook the *app* function in Section 5.1 has a slight typo. Enter the function into your file. If there is a typo, fix it. (If the function compiles and works as is, then proceed to the next problem.)

```
val app : ys:'a list -> zs:'a list -> 'a list
```

```
app [1;2;3][4;5;6];;
val it : int list = [1; 2; 3; 4; 5; 6]
```

Problem 3 (Lists – 3 points) Use List.foldBack to write a function *doubleList* that returns a list concatenated with itself.

```
val doubleList : list:'a list -> 'a list
```

```
doubleList [1..4];;
val it : int list = [1; 2; 3; 4; 1; 2; 3; 4]
```

Problem 4 (Lists – 3 points) Use List.fold to write a function *backwardsAndForwardsList* that returns the reverse of a list concatenated with the list itself.

```
val backwardsAndForwardsList : list:'a list -> 'a list
```

```
backwardsAndForwardsList [1..4];;
val it : int list = [4; 3; 2; 1; 1; 2; 3; 4]
```

Problem 5 (Lists – 4 points) A polynomial may be represented as a list of coefficients and exponents. For example, the polynomial $f(x) = x^2 + 2x + 5$ can be represented as [(1.0,2);(2.0,1);(5.0,0)]. Add to your program a corresponding *Polynomial* type:

```
type Polynomial = (float * int) list
```

Then define an *evaluate* function that evaluates a polynomial at a float value x. For instance, if f is the polynomial given above, then we have f(3.0) = 20.0.

```
val evaluate : poly:Polynomial -> x:float -> float
```

```
evaluate [(1.0,2);(2.0,1);(5.0,0)] 3.0;;
val it : float = 20.0
```

```
evaluate [(3.5,4);(-3.0,0)] 2.1;;
val it : float = 65.06835
```

```
evaluate [] 4.0;;
val it : float = 0.0
```

Problem 6 (Sets – 3 points) This problem is adapted from Problem 5.6.1 in the textbook. A *relation* from a set A to a set B is a set of ordered pairs of the form *(a,b)* where $a \in A$ and $b \in B$. (In other words, a relation from A to B is a subset of $A \times B$.) If relation *r* is *finite* then it can be represented in F# as a set of pairs. The domain *dom r* is the set of elements $a \in A$ such that for some $b \in B$, $(a, b) \in r$. Use Set.fold to define *dom* as an F# function.

```
val dom : s:Set<'a * 'b> -> Set<'a> when 'a : comparison and 'b : comparison
```

```
dom (set [("Jill",78);("Joe",77);("Joe",67);("Suzi",66);("Zeke",67);("Jill",84)]);;
val it : Set<string> = set ["Jill"; "Joe"; "Suzi"; "Zeke"]
```

```
dom (Set.empty : ((int * float) Set));;
val it : Set<int> = set []
```

Problem 7 (Maps and Sets – 3 points) Use the Map.fold to write a function *mapDom* that computes the domain of a Map. (This problem is very similar to the preceding problem.)

```
val mapDom : m:Map<'a,'b> -> Set<'a> when 'a : comparison
```

```
mapDom (Map.ofList [("Jack",11);("Jill",12);("Hill",23)]);;
val it : Set<string> = set ["Hill"; "Jack"; "Jill"]
```

Problem 8 (Maps – 3 points) Use Map.fold to write a function *sumOfRangeElts* that takes a map whose range values are ints and returns the sum of the range values.

```
val sumOfRangeElts : m:Map<'a,int> -> int when 'a : comparison
```

```
sumOfRangeElts (Map.ofList [("Jack",11);("Jill",12);("Hill",23)]);;
val it : int = 46
```

Problem 9 (Maps – 3 points) Use Map.map to write a function *prefixKeysToValues* that takes a map of strings to integers and returns a new map that is the same as the original except that, in each entry, the value is prepended with the key. If necessary, review the *toString* example in Section 3.7 to see how to convert ints to strings and how to concatenate strings.

```
val prefixKeysToValues : m:Map<string,int> -> Map<string,string>
```

```
prefixKeysToValues (Map.ofList [("Jack",11);("Jill",12);("Hill",23)]);;
val it : Map<string,string> =
  map [("Hill", "Hill23"); ("Jack", "Jack11"); ("Jill", "Jill12")]
```

Problem 10 (Maps – 4 points) Use Map.fold to write a *compose* function. If m1 and m2 are maps then *compose m1 m2* is the composition of m1 and m2. I.e., *compose m1 m* is the map that consists of the entries (a,c) such that there is a value b where (a,b) is an entry in m1 and (b,c) is an entry in m2. The body of the instructor's sample solution for this problems is four lines long and also uses Map.containsKey, Map.add, Map.find, and Map.empty. (Map.empty is an empty map.) Hint: For each entry (m1x, m1y) in m1, check whether m2 contains m1y as a key. If it does, then add the appropriate entry to the result.

```
val compose :
  m1:Map<'a,'b> -> m2:Map<'b,'c> -> Map<'a,'c>
    when 'a : comparison and 'b : comparison
```

```
compose (Map.add 2 5 (Map.add 1 3 (Map.add 0 2 Map.empty)))
             (Map.add 5 "five" (Map.add 3 "three" (Map.add 2 "two"
                  Map.empty)));;
val it : Map<int,string> = map [(0, "two"); (1, "three"); (2, "five")]
```

```
compose (Map.ofList [(333,"Jill");(222,"Hill");(111,"Jack");(444,"Jill")])
             (Map.ofList [("Jack",32.7);("Jill",31.6);("Fill",46.2)]);;
val it : Map<int,float> = map [(111, 32.7); (333, 31.6); (444, 31.6)]
```

How to submit your code for automated grading:

Once you have tested your code and believe it to be correct, go to our course on Canvas and visit "Assignments"->"HW4: F# collections". Then click the "Load HW4: F# collections" button at the bottom of the page. This will take you to an automated grading program called Mimir. Upload your HW4.fs file there. Mimir will run it the test cases from HW4Test.fs and show your score. If your program fails any test cases then you can click on those cases to see the problem. You will be able to see which input Mimir provided to HW4Test.fs, what your program's output was, and what the correct output is. From that point you can revise your program and resubmit, if necessary.

My final checks of your code:

Within a week or so of the assignment deadline I will take a look at the code that you submitted through Mimir to check that you followed the directions. For example, in Problem 1, did you actually use List.fold in your solution? Once I have checked everyone's code I will release the grades to Canvas.