CPSC 3400 Languages and Computation
Winter 2024

Homework 4
Due: Monday, February 12 at 10:00pm

This assignment consists of three F# exercises. The code for all exercises are to be included in a single file named `hw4.fsx`.

1. `replace list origVal newVal`: The function takes a list of integers (`list`), an integer to be replaced (`origVal`), and a replacement integer (`newVal`). The function returns a list such that each occurrence of `origVal` in `list` is replaced with `newVal`. All other elements remain unchanged.

Examples:

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

2. `mergeList listA listB`: Both parameters are lists of integers. The function returns a list that merges the lists in alternating fashion. The first item in the first item from `listA`, the second item is the first item from `listB`, the third item is the second item form `listA`, and so on. Continue in this fashion until one list runs out; then append the remaining items from the longer list.

Examples:

```
> let m1 = [1; 2; 3; 4; 5; 6];;
> let m2 = [7; 8; 9; 10];;
> mergeList m1 m2;;
val it : int list = [1; 7; 2; 8; 3; 9; 4; 10; 5; 6]
```

3. Write the following functions for a dictionary. The dictionary is implemented using a list of tuples. Each tuple represent a key-value pair has two items: the first is a string and represents the key, the second is an integer and represents the value. The dictionary is unordered and each key must be unique.

a. `search dict key`: Returns an option type with the corresponding value is if the key is in the dictionary and `None` if the key is not in the dictionary.

b. `insert dict key value`: If the key is not already in the dictionary, a new dictionary is returned with the key-value pair inserted into the dictionary. If the key is already in the dictionary, return the dictionary with no modifications.

c. `remove dict key`: If the key is in the dictionary, a new dictionary is returned with the corresponding key-value pair removed. If the key is not in the dictionary, return the dictionary with no modifications.

d. `count dict func`: The parameter `func` is a Boolean function that takes a single integer parameter and returns true or false. The function calls `func` with each value (of the key-value pair) and returns the number of nodes that evaluate to true.

e. `twoDigitCount dict`: Returns the number of entries that have a two-digit positive integer (10 to 99) value. REQUIREMENT: This function must be a single call to `count` using a lambda function.

Examples:

```
> let fruits = [("apple", 5); ("pear", 73); ("lime", 42); ("orange", 8);
("kiwi", 17)];;

> search fruits "pear";;
val it : int option = Some 73
> search fruits "banana";;
val it : int option = None

> insert fruits "pear" 16;;
val it : (string * int) list =
  [("apple", 5); ("pear", 73); ("lime", 42); ("orange", 8); ("kiwi", 17)]
> insert fruits "banana" 44;;
val it : (string * int) list =
  [("banana", 44); ("apple", 5); ("pear", 73); ("lime", 42); ("orange", 8);
   ("kiwi", 17)]

> remove fruits "pear";;
val it : (string * int) list =
  [("apple", 5); ("lime", 42); ("orange", 8); ("kiwi", 17)]
> remove fruits "banana";;
val it : (string * int) list =
  [("apple", 5); ("pear", 73); ("lime", 42); ("orange", 8); ("kiwi", 17)]

> let countAll x = true;;
val countAll : x:'a -> bool
> let countEven x = x % 2 = 0;;
val countEven : x:int -> bool
> count fruits countAll;;
val it : int = 5
> count fruits countEven;;
val it : int = 2

> twoDigitCount fruits;;
val it : int = 3
```

Implementation Notes:

- You may define and use additional functions if your feel it is appropriate.
- Functions must match the order of parameters given.
- As long as the functions accept the specified types, it is permissible for the functions to be generic and accept other types. In other words, you should not have to specify the types of any parameters unless it is necessary for correct functionality.
- Without prior approval from me, you may only use the subset of F# described in class.
    - In particular, you may NOT use F# system functions or methods (especially those in the map module).
- You are only required to write the provided functions. There is no additional global functionality. Your program should produce no output and accept no input.
- To test your functions, either:
    - Test the functions using an interactive interpreter OR
    - Add your own tests to the script (but be sure to remove them before submission!)

Submitting your Assignment

To submit your assignment, you will need to transfer your file to `cs1`. On `cs1`, run the following script in the directory with your program:

```
/home/fac/bdiazacosta/submit/cpsc3400/hw4_submit
```

This will copy the files `hw4.fsx` to a directory that can be accessed by the instructor. Please be sure to keep the same file name or the submission program will not work. Only the last assignment submitted before the due date and time will be graded. ***Late submissions are not accepted and result in a zero.***

Grading

Grading is based on the following rubric and is primarily based on functionality based on how many tests pass or fail.

Feature	Max Points
`replace` • Provided example: 2 points • 6 additional tests: 1 point each	8
`mergeList` • Provided example: 2 points • 6 additional tests: 1 point each	8
`search` • 2 provided examples: 2 points each • 3 additional tests: 1 point each	7
`insert` • 2 provided examples: 2 points each • 3 additional tests: 1 point each	7
`remove` • 2 provided examples: 2 points each • 3 additional tests: 1 point each	7
`count` • 2 provided examples: 2 points each • 3 additional tests: 1 point each	7
`twoDigitCount` • Provided examples: 2 points • One additional test: 1 point • Properly implemented using a single call to count with a lambda function: 3 points	6
TOTAL	**50**

The following additional deductions are possible:

- Programs may lose up to 10 points if they exhibit poor readability or style.
- Using F# constructs or system functions outside the subset in class will result in a deduction. Severe deductions are possible if the use significantly simplifies one of the functions and/or results in a style that is not pure functional programming. I also reserve the right to require a resubmission.
- Programs that contain syntax errors will receive a zero.