CSE 341 Sample Midterm #1

1. Expressions

For each ML expression in the left-hand column of the table below, indicate in the right-hand column its value. Be sure to (e.g., `7.0` rather than `7` for a `real`; `String`s in quotes e.g. `"hello"`; `true` or `false` for a `bool`). Assume that the following value has been declared:

```
val numbers = [3, 4, 9];
```

Expression	Value
a) `reduce(op +, numbers)`	
b) `mapx(fn x => x--(x+1), numbers)`	
c) `mapx(length o tl, [1--5, 3--5, 5--5, 2--5])`	
d) `filterx(fn x => 24 mod x = 0, 1--7)`	
e) `reduce(op *, mapx(fn x => x - 1, numbers))`	
f) `mapx(fn x => (x, x), numbers)`	
g) `reduce(op +, mapx(fn x => 2 * x - 1, 1--5))`	
h) `reduce(op @, mapx(fn x => [x, x], numbers))`	
i) `filterx(fn x => x div 4 = 3, 1--15)`	
j) `implode(mapx(hd o rev o explode, ["foo","bar","baz"]))`	

2. Types

For each ML expression in the left-hand column of the table below, indicate its **type** in the right-hand column assuming the expression was added to the top-level environment. Assume that the following value has been declared:

```
val lst = ["to", "be", "or", "not", "to", "be"];
```

Expression	Type
a) `lst`	
b) `(lst, lst)`	
c) `[lst, lst]`	
d) `fn x => (x, [[x + 1], [x + 2]])`	
e) `fn (x, y) => [(x, 1), (2, y)]`	

3. Scope

What value does the variable `answer` store after executing the following code?

```
val x = 40;
val y = 5;
val z = 25;
fun f(x) =
    let val z =
            let val y = 2
                val x = 10
            in  x + y
            end
        val y = x - 5
        val x = y - 5
    in x + y + z
    end
val answer = f(20);
```

CSE 341 Sample Midterm #1

4. Curried Functions

For this problem, in addition to being able to call the functions listed on the front page of the test, you may call the function `curry` and the curried versions of `map`, `filter` and `List.foldl/r`) that we used in Homework #3, as well as the following curried functions:

- `fun curry f x y`
  Returns a curried version of the 2-argument function/operator `f`
- `fun map2 f list`
  curried version of `map` written in lecture
- `fun filter2 f list`
  curried version of `filter` written in lecture
- `fun reduce2 f list`
  curried version of `reduce` written in lecture

As in homework #3, you may use only `val` definitions to solve the following problems. You may not use `fun` or `fn` definitions to define a function.

a) Define a function `product` that takes a list of integers as an argument and that returns the product of the *nonzero* values in the list. For example, `product([2, 0, 3, 0, 4])` should return 24. You may assume there is at least one nonzero value in the list.

b) Define a function `starString` that takes a list of strings as an argument and that returns a new string obtained by concatenating the strings together with an asterisk in front of each. You may assume the list has at least one string. For example:

- `starString(["one"])` should return `"*one"`
- `starString(["foo", "bar", "baz"])` should return `"*foo*bar*baz"`
- `starString(["a", "b", "c", "d"])` should return `"*a*b*c*d"`

CSE 341 Sample Midterm #1

5. Functions

Write a function `zip` that takes two lists as arguments and that returns the list obtained by combining pairs of values in corresponding positions into tuples. The first tuple should contain the first values from each list. The second tuple should contain the second values from each list. And so on. If one list is shorter than the other, then you should return a list of that shorter length. For example:

- `zip([1, 2, 3], ["a","b","c"])` should return `[(1,"a"),(2,"b"),(3,"c")]`
- `zip([1, 2, 3, 4, 5], ["a","b","c","d"])` should return `[(1,"a"),(2,"b"),(3,"c"),(4,"d")]`
- `zip(["a", "b", "c", "d"], [1, 2])` should return `[("a",1),("b",2)]`

6. Functions

Write a function `digitSum` that takes an integer n as an argument and that returns the sum of the digits of n. For example:

- `digitSum(384)` should return 15
- `digitSum(333)` should return 9
- `digitSum(~71)` should return 8
- `digitSum(0)` should return 0
- `digitSum(208034239)` should return 31

7. Functions

Write a function `maxSizeSublist` that takes a list of strings and an integer n as arguments and that returns a list of n of those strings whose sizes are maximal. In other words, the function is picking the n elements of the list that have the greatest size. Any given string can be used only once. Assume that the list contains no duplicates. The list you return can be in any order and if there is more than one list of maximal size, then you can return any one of them. For example:

- `maxSizeSublist(["four", "score", "and", "seven", "years", "ago"], 1)` could return `["seven"]`
- `maxSizeSublist(["four", "score", "and", "seven", "years", "ago"], 3)` could return `["seven","score","years"]`
- `maxSizeSublist(["aaaa", "b", "cc", "ddd", "eeeee"], 3)` could return `["eeeee","aaaa","ddd"]`

Recall that the function called `size` is used to find the size of a string. You may assume that the parameter n is greater than or equal to 0 and that it is less than or equal to the length of the list.

8. Datatypes

Recall the `IntTree` data type we discussed in lecture for storing a binary tree of integers:

```
datatype IntTree = Empty | Node of int * IntTree * IntTree;
```

a) Write a function `sameStructure` that takes two `IntTree`s as arguments and that returns whether or not the two trees have the same structure (true if they do, false otherwise). Two trees are considered to have the same structure if they have the same number of nodes in the same relative positions (the data stored in the tree doesn't matter).

b) Write a function `atLevel` that takes an `IntTree` and an integer n as arguments and that returns a list of the data values stored at level n in the tree. The overall root is considered to be at level 1, its children are at level 2, its grandchildren are at level 3, and so on. The values should be listed from left to right (i.e., in the same order in which they would be visited by any standard tree traversal). You may assume that n is greater than or equal to 1.

CSE 341 Sample Midterm #1

9. Functions

Write a function `sameDigitSum` that takes two lists of integers and that returns a list of all (x, y) tuples where x is from the first list, y is from the second list, both are in the same position in their respective lists, and both have the same digit sum. For example:

- `sameDigitSum([13, 8, 72, 308], [202, 13, 308, 2020403])`
  should return `[(13,202),(308,2020403)]`

The first pair of numbers has a digit sum of 4 and the second pair of numbers has a digit sum of 11. Notice that we don't get (13, 13) as an answer even though the number 13 appears in both lists. That match doesn't appear because the two occurrences of 13 are in different positions in the list. Also notice that the pairs should be listed in the order in which they appear in the original lists.

Notice that one list can be shorter than the other. For example:

- `sameDigitSum([12, 42, 73], [111, 55, 82, 19, 201, 24])` should return `[(12,111),(73,82)]`

10. Functions

Write a function `combine` that takes a list of integers and a minimum value n as arguments and that returns a list of tuples formed by combining sequences of sequential values from the list so that their sum is greater than or equal n. Your function should combine values from the front of the list by adding them up until their sum is at least n. Once you have a sum that is at least n, you should include a tuple in the result that includes the count of values combined and the sum. The function should then combine values again until it finds a sum that is at least n and should include a count/sum tuple for that group of combined values. It should continue combining values in this way until it reaches the end of the list. For example:

- `combine([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], 10)`
  should return `[(4,10),(2,11),(2,15),(2,19),(1,11),(1,12)]`

This result indicates that the first 4 values from the list were combined for a sum of 10 (the values 1, 2, 3, and 4), then 2 values were combined for a sum of 11 (the values 5 and 6), then 2 values were combined for a sum of 15 (the values 7 and 8), then 2 values were combined for a sum of 19 (the values 9 and 10), then 1 value was combined for a sum of 11 (the value 11), and finally 1 value was combined for a sum of 12 (the value 12).

It is possible that your function will run out of values to combine at the end of the list, so that the final sum might not be greater than or equal to n. That is okay. For example:

- `combine([3, 8, 7, 6, 12, 2, 4, 7, 2, 9, 5, 2], 11)`
  should return `[(2,11),(2,13),(1,12),(3,13),(2,11),(2,7)]`

Notice that the final tuple has a sum of 7, which is less than the 11 minimum we are trying to achieve. This should happen only for the last sequence of numbers combined. You may assume that all of the numbers in the list are positive. If there are no values to combine, your function should return an empty list.

CSE 341 Sample Midterm #1
Solutions

1. Expressions

Expression	Value
a) `reduce(op +, numbers)`	`16`
b) `mapx(fn x => x--(x+1), numbers)`	`[[3,4],[4,5],[9,10]]`
c) `mapx(length o tl, [1--5, 3--5, 5--5, 2--5])`	`[4,2,0,3]`
d) `filterx(fn x => 24 mod x = 0, 1--7)`	`[1,2,3,4,6]`
e) `reduce(op *, mapx(fn x => x - 1, numbers))`	`48`
f) `mapx(fn x => (x, x), numbers)`	`[(3,3),(4,4),(9,9)]`
g) `reduce(op +, mapx(fn x => 2 * x - 1, 1--5))`	`25`
h) `reduce(op @, mapx(fn x => [x, x], numbers))`	`[3,3,4,4,9,9]`
i) `filterx(fn x => x div 4 = 3, 1--15)`	`[12,13,14,15]`
j) `implode(mapx(hd o rev o explode, ["foo","bar","baz"]))`	`"orz"`

2. Types

Expression	Type
a) `lst`	`string list`
b) `(lst, lst)`	`string list * string list`
c) `[lst, lst]`	`string list list`
d) `fn x => (x, [[x + 1], [x + 2]])`	`int -> int * int list list`
e) `fn (x, y) => [(x, 1), (2, y)]`	`int * int -> (int * int) list`

3. Scope

```
val answer = 37
```

4. Curried Functions

```
val product = reduce2 op* o (filter2 (curry op<> 0));
val starString = reduce2 op^ o (map2 (curry op^ "*"));
```

5. Functions

```
fun zip(_, []) = []
|   zip([], _) = []
|   zip(x::xs, y::ys) = (x, y)::zip(xs, ys);
```

6. Functions

```
fun digitSum(n) =
    if n < 0 then digitSum(~n)
    else if n < 10 then n
    else digitSum(n div 10) + n mod 10;
```

CSE 341 Sample Midterm #1
Solutions (continued)

7. Functions

```
fun maxSizeSublist(lst, n) =
    let fun helper(_, 0) = []
        |   helper(x::xs, m) = x::helper(xs, m - 1)
    in helper(quicksort(fn (x, y) => size(x) >= size(y), lst), n)
    end;
```

8. Data Types

```
a)
fun sameStructure(Empty, Empty) = true
|   sameStructure(Empty, Node(_, left, right)) = false
|   sameStructure(Node(_, left, right), Empty) = false
|   sameStructure(Node(_, left1, right1), Node(_, left2, right2)) =
    sameStructure(left1, left2) andalso sameStructure(right1, right2);

b)
fun atLevel(Empty, _) = []
|   atLevel(Node(root, left, right), 1) = [root]
|   atLevel(Node(_, left, right), n) =
        atLevel(left, n - 1) @ atLevel(right, n - 1);
```

9. Functions

```
fun sameDigitSum(lst1, lst2) =
    filterx(fn (x, y) => digitSum(x) = digitSum(y), zip(lst1, lst2));
```

10. Functions

```
fun combine([], _) = []
|   combine(x::xs, min) =
        let fun process([], sum, count) = [(count, sum)]
        |   process(x::xs, sum, count) =
                if sum < min then process(xs, sum + x, count + 1)
            else (count, sum)::process(xs, x, 1)
    in process(xs, x, 1)
    end;
```