Tutorial 3

EECS-111 Fall 2015

Part 1: Data flow diagrams (pencil and paper)

Draw the dataflow diagrams for the following expressions (answers are given at the end of this file).

1. (+ 1 1)

2. (filter odd?
       (map abs
           (list 1 -2 3)))

3. (foldl +
       0
       (list 1 2 3))

4. (list 1 2 3 4)

Part 2: Simple list operations (using DrRacket)

1. Make a list containing the numbers 1 2 and 4, and the string "I am the walrus". That is, run code in DrRacket that makes that list.

2. Now define a variable, **mylist**, to be a list of numbers. Use whatever numbers you like.

3. Now write two different expressions that compute the **sum** of all the numbers in mylist.

4. Now write an expression that **triples** all the numbers in the list. The result should be a new list of numbers.

5. Now write an expression that finds all the numbers in mylist that are **between 5 and 20**. Note that to properly test this, the list needs to contain both numbers in and outside the range. So if there aren't any numbers in that range, add some. If there are only numbers in that range, add some that aren't.

6. Suppose you have a list of lists of numbers:

   (define listoflists

```
(list (list 1 2 3)
      (list 4 5 6)
      (list 1 1 1 1)))
```

Write an expression to compute the **sum of all the numbers in all the sublists** (hint: make a list of the sums of each of the sublists, then add that up).

Part 3: Social networks (DrRacket)

For this part, **open the file Tutorial 3.rkt and run it**.

Suppose you're working at a new startup, Grimacebook, that's building a new social networking system. Like every other social networking system, it has to keep track of who is friends with who. So it has a big list of friends:

```
(define friends-db
 (list (list "ben" "jerry")
       (list "martha" "paul")
       (list "hillary" "bernie")
       (list "lizbeth" "mikael")
       (list "edward" "bernie")
       (list "steve" "hillary")
       (list "larry" "edward")
       (list "sheryl" "hillary")
       (list "woz" "lizbeth")
       (list "lizbeth" "edward")
       (list "elliot" "lizbeth")
       (list "edward" "elliot")
       (list "lizbeth" "berkoff")
       (list "berkoff" "nikita")))
```

Each element of the list is itself a two element list stating two users who are friends. Note that the friends are written in no particular order, so if Sally is a friend of Jane, it might appear in the list as (list "sally" "jane") or as (list "jane" "sally").

1. Write a procedure, **friends?**, that takes the names of two users and returns true if they're friends. Make sure that it works no matter what order the arguments are given in. That is, if Sally and Jane are friends, both (friends? "sally" "jane") and (friends? "jane" "sally") should work.

   Hint: Since friends-db is a list of two-element lists, each of which expresses two friends, you want to start by writing a procedure that tests a two-element list to see if it's listing the specific pair of friends you're looking for. Having done that, you can use **filter** or **ormap** to search friends-db for the list you're looking for.

2. Now write an expression that computes **all the users in the database**. That is, scan through the database and make one flat list of all the users. So your answer should look something like (list “ben” “jerry” “martha" “paul” …). Don’t worry about the order they appear in the list. There are lots of different ways of doing this. One simple way is to first append all the sublists together into one big list, and then remove the duplications using remove-duplicates.

   Put your list of all the users in the variable **all-users**.

3. Now write a procedure, **friends-of**, that returns a list of all the users that are friends of a given user. So (friends-of “ben”) would return (list “jerry”), but (friends-of “hillary") would return (list “bernie" “steve” “sheryl”). Again, don’t worry about the order in which the friends are listed. Again, there are many ways of doing this.

4. Now write a procedure, **popularity**, that tells you how many friends a user has. So (popularity “ben”) would return 1 and (popularity “hillary") would return 3.

5. Write a procedure, **mutual-friends**, that takes two users as arguments, and returns the friends they have in common. If they have no friends in common, it should return the empty list.

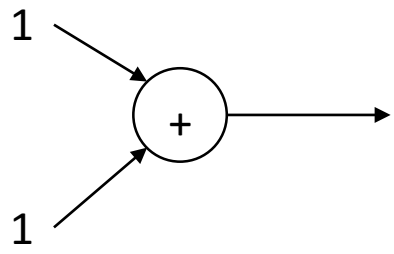
   Hint: find the lists of friends for each user. Then filter one list for just the elements that appear in the other list. You can ask if an element appears in a list using the **member** function: (member x list) is true if and only if x appears in list.

6. Write an expression to determine which user is **most popular**.

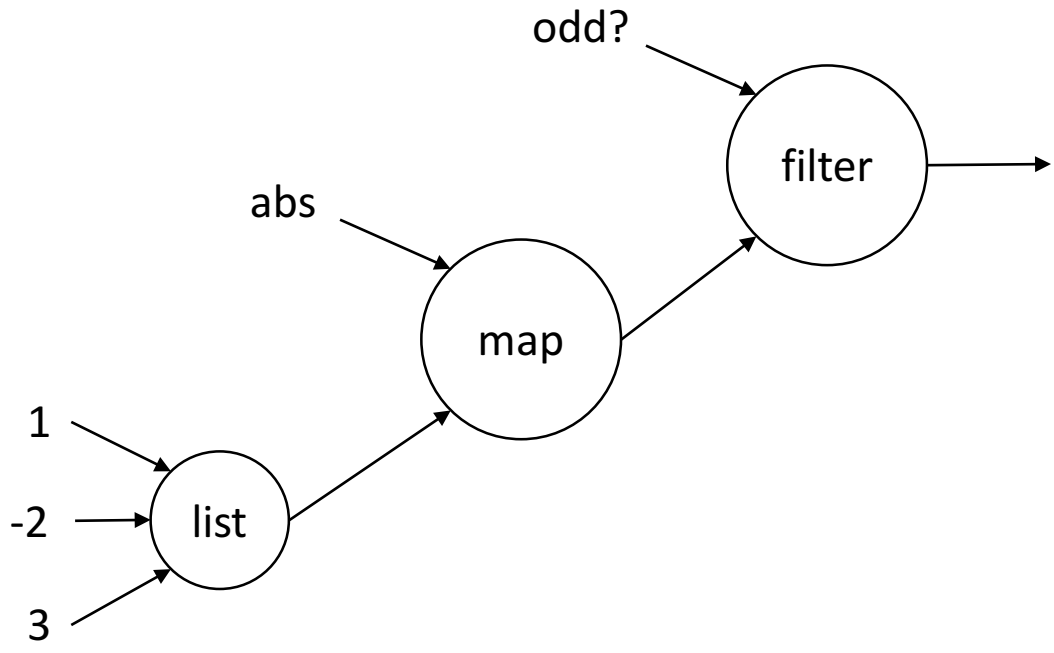
Solutions to part 1

Draw the dataflow diagrams for the following expressions (answers are given at the end of this file).

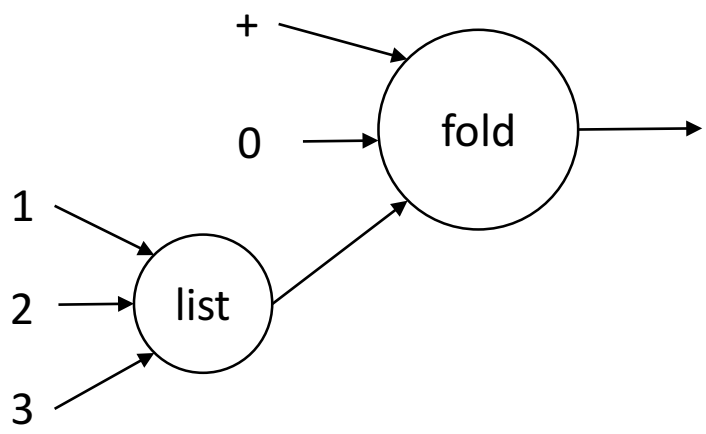
1. (+ 1 1)



2. (filter odd?
   (map abs
   (list 1 -2 3)))



3. (foldl +
   0
   (list 1 2 3))



4. (list 1 2 3 4)