**Lab # 1**

Due: 2024-09-27

By: Omar Abdul, Anas Taimah, Yusuf Khan

300228700, 300228842, 300293842

Course: CSI3120

Group: 40

**Code explanation:**

Question 1:

The function map2 takes the input lst1 and lst2 as well as a function f (+, -, %, etc.) that applies to the items in the list. In the base case, if both lists are empty, it returns an empty list. In the second case if both lists are greater than size 0, it takes the head of each list and defines the tail of each list “(x::xs, y::ys)” and applies the function to it “(f x y)” to create the head of a new list being returned. The tail of this new list is then recursively defined by calling the map2 function on the tail of the list until it is empty. The third case of this function happens when the lists are of two different lengths.

Question 2:

This function filter\_even takes a list as an input and filters out all the odd numbers from it. It uses List.filler which is a built in function that takes a function and the list. It goes through each element x in the list and applies the predicate (fun x -> x mod 2 = 0) to check if x is divisible by 2. If the predicate returns true, x is kept in the list and if the predicate returns false, x is removed from the list.

Question 3:

The first part is the function compose\_functions which takes 2 functions as an argument. The function then defines a new function “f (g(x))” which applies function g on input x and then applies function f to that result.

The second part is creating a variable called composed which calls the compose\_functions on an input with two defined functions f (fun x -> x \* 2) & g (fun y -> y + 3). The defined function f multiplies by 2 and the defined function g adds 3.

When combined it means the input is first incremented by 3 and then multiplied by 2 to give us our final result. Composed 5 = 2(5 + 3) = 16

Question 4:

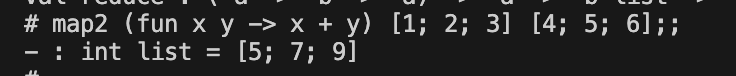
A recursive function “reduce” is defined which takes 3 inputs (a function, an accumulator, a list). The base case checks if the list input is empty which returns the accumulator value. The second case when the list is not empty splits the list into head x and tail xs, and then calls the reduce function recursively with three new inputs; 1) same function 2) a new accumulator which is calculated by applying the function on the accumulator and the head 3) the tail of the list.

Once all the recursive calls are done and the tail list is empty, the accumulator is returned.

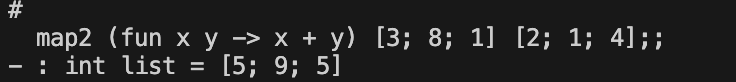
**Test Cases:**

Question 1:

Test 1:

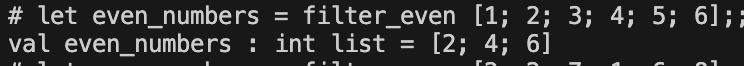


Test 2:

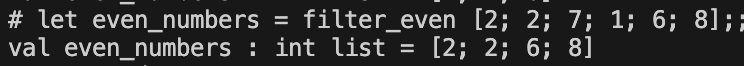


Question 2:

Test 1:



Test 2:



Question 3:

Test 1:

A black background with white text

Description automatically generated

Test 2:

A black background with white text

Description automatically generated

Question 4:

Test 1:



Test 2:



All Test Cases Screenshot:

A computer screen shot of a program

Description automatically generated