

[Functional] Programming Exercises

CS 364 — Spring 2022

1 Definitions and Background

1. Define the following terms and give examples where appropriate.

(a) binding:

(b) lambda expression:

(c) variant type:

(d) first class function:

(e) higher-order function:

(f) closure:

(g) referential transparency:

2. What are some differences between programming languages? Provide several concrete examples.

3. Briefly describe Imperative, Object-Oriented, Functional, and Declarative programming paradigms. What some typical characteristics of each?

2 Recursion

1. What is tail recursion? Why is it desirable?

2. Rewrite the following function such that it is tail-recursive.

```
let rec fib = (n:int) : int => {
  if( n < 0 ) {
    failwith("negative_input_is_not_allowed");
  } else {
    if( n == 0 || n == 1 ) {
      1;
    } else {
      fib( n - 1 ) + fib( n - 2 );
    }
  }
};
```

3. Write a recursive function called power that inputs two non-negative integers x and y and outputs x^y using multiplication.

```
let power = ( (x:int), (y:int) ) : int =>
```

3 Function Evaluation

Evaluate the following expressions, showing several steps on the way to the final value.

1. $((x, y) \Rightarrow \text{abs}(x - y))(4, 8)$;

2. `List.filter (x => { x mod 2 == 0 },
List.map (x => { x + 3 }, [1, 2, 4, 5, 6, 10]))`;

```

3. let rec fold = ( (f : ('b, 'a) => 'b), (acc : 'b), (lst : list('a)) ) : 'b => {
    switch( lst ) {
      | [] => acc
      | [hd, ...tl] => fold( f, f( acc, hd ), tl )
    };
  };
fold ( ( pred, a ) => { pred || a > 5 }, false, [ 0, 3, 2, -1, 6 ] );

```

4 Higher-Order Functions

Consider the following function definition for `fold2`, which folds over two, equal-length lists:

```

let rec fold2 = ( (f: ('a, 'b, 'c) => 'a), (acc:'a), (l1:list('b)), (l2:list('c')) ) : 'a => {
  switch( (l1, l2) ) {
    | ([],[ ]) => acc
    | ([hd1, ...tl1], [hd2, ...tl2]) => fold2( f, f( acc, hd1, hd2 ), tl1, tl2 )
    | _ => failwith("lists_have_different_lengths")
  };
};

```

This function can be used to implement other higher-order functions. Demonstrate this ability by implementing the following functions using `fold2`.

```

1. /*
   * Given f, [a1, ..., an], [b1, ..., bn]
   * return [ f(a1, b1), ..., f(an, bn) ]
   */
let map2 = ( (f: ('a, 'b) => 'c), (l1: list('a)), (l2: list('b)) ) : list('c) =>

```

```
2. /*
   * Given f, [a1, ..., an], [b1, ..., bn]
   * return true if f(ai, bi) returns true for all 1 <= i <= n
   */
let for_all2 = ( (f: 'a => 'b => bool), (l1: list('a)), (l2: list('b)) ) : bool =>
```