COL100: Lab 4 Solutions

In case of any error please contact Praveen Kulkarni at `cs5140599@cse.iitd.ac.in`.

collatz.ml

```ocaml
(* Author: Praveen Kulkarni
 * Date: 14 March 2018
 * File: collatz.ml
 * All rights reserved. Copyright (c) 2018
 *)

(* collatz : int -> int *)
let rec collatz x =
    if x = 1 then 0
    else if x mod 2 = 0 then 1 + collatz (x/2)
    else 1 + collatz (3*x + 1);;

(* test cases *)
let _ = collatz 1;;
let _ = collatz 12;;
let _ = collatz 19;;
let _ = collatz 27;;

(*
 * bestx is the value of x such that collatz x is the largest
 * amongst all the values of x that we have seen so far. maxlen
 * is the corresponding value of collatz x.
 * Function tries out the x's in a descending order.
 *)
(* f : int -> int -> int -> int *)
let rec f (x) (bestx) (maxlen) =
    if x = 0 then bestx
    else
        let currlen = collatz x in
            if currlen > maxlen then f (x-1) (x) (currlen)
            else f (x-1) (bestx) (maxlen);;

(* max_collatz : int -> int *)
let max_collatz x =
    if x = 1 then 1
    else (f (x) (x) (collatz x));;

(* test cases *)
let _ = max_collatz 10;;
let _ = max_collatz 100;;
```

hanoi.ml

```ocaml
(* Author: Praveen Kulkarni
 * Date: 14 March 2018
 * File: hanoi.ml
 * All rights reserved. Copyright (c) 2018
 *)

(* The three rods are called the
 * `src` : source rod
 * `dst` : destination rod
 * `aux` : auxiliary rod
 *
 * The logic of the solution is explained here:
 * https://en.wikipedia.org/wiki/Tower_of_Hanoi
 *)

(* hanoi1 : int -> int -> int -> int -> string *)
let rec hanoi1 (n) (src) (dst) (aux) =
    let motion = "(" ^ (string_of_int src) ^ ", " ^ (string_of_int dst) ^ ")\n" in
        if n = 1 then motion
        else
            let prefix = hanoi1 (n-1) (src) (aux) (dst) in
            let suffix = hanoi1 (n-1) (aux) (dst) (src) in
            prefix ^ motion ^ suffix;;

(* hanoi : int -> string *)
let hanoi n = hanoi1 n 1 3 2;;

(* test cases *)
print_string(hanoi 2);;
print_string(hanoi 3);;
```

josephus.ml

```ocaml
(* Author: Praveen Kulkarni
 * Date: 14 March 2018
 * File: josephus.ml
 * All rights reserved. Copyright (c) 2018
 *)

(* This is a difficult recursion problem. Find a discussion of the
 * problem here :
 * https://en.wikipedia.org/wiki/Josephus_problem#The_general_case
 *)

(* Assume that the starting position is 1 *)
(* josephus1 : int -> int -> int *)
let rec josephus1 (n) (k) =
    if n = 1 then 1 (* only one person remains, so he survives *)
    else (((josephus1 (n-1) k) + k - 1) mod n) + 1;;

(* cyclic shift to start from `start`. *)
(* josephus: int -> int -> int -> int *)
let josephus (n) (k) (start) =
    let position1 = josephus1 (n) (k) in
    ((position1 - 1 + (start - 1)) mod n)+ 1;;

(* test cases *)
let _ = josephus 5 3 1;;
let _ = josephus 7 4 2;;
```