```
(* week-03_exercises.ml *)
(* Introduction to Computer Science (YSC1212), Sem2, 2021-2022 *)
(* Olivier Danvy <danvy@yale-nus.edu.sg> *)
(* Version of Sat 29 Jan 2022 *)

(* ********** *)

(*
Name of the group on Canvas: WEEK-03_4
*)

(* ********** *)

(* Exercise 13 *)

let exercise_13 = "mandatory";;

let test_successor candidate = (candidate 0 = 0+1)
&& (candidate 1 = 1+1)
&& (candidate 2 = 2+1)
&& (candidate 3 = 3+1)
&& (candidate 4 = 4+1)
&& (candidate 5 = 5+1);;

let exercise_13_positive_test_a = (test_successor (fun i -> i + 1) = true);;

let exercise_13_positive_test_b = (test_successor (fun i -> 1 + i) = true);;

let exercise_13_positive_test_c = (test_successor succ = true);;


let exercise_13_negative_test_a = (test_successor (fun i -> i) = false);;

let exercise_13_negative_test_b = (test_successor (fun i -> i + 2) = false);;

let exercise_13_negative_test_c = (test_successor (fun i -> 2 * i) = false);;


(* Can you implement a fake successor function,
   i.e., one that is incorrect but that passes the unit tests? *)

let fake_successor n = abs(n) + 1;;

let exercise_13_test_fake_successor = (test_successor fake_successor = true);;


(* ********** *)


(* Exercise 14 *)

let exercise_14 = "mandatory";;

let test_not candidate =
  (candidate true = false)
&& (candidate false = true);;

let exercise_14_positive_test_a = (test_not not = true);;
```

```
let exercise_14_negative_test_a = (test_not (fun b -> b) = false);;

let neg_v1 b =
  if true then false else true;;

let exercise_14_positive_test_b = (test_not neg_v1 = false);;

let neg_v2 b =
  b = false;;

let exercise_14_positive_test_b = (test_not neg_v2 = true);;


(* Can you implement a fake negation function,
   i.e., one that is incorrect but that passes the unit tests? *)
(*
let fake_not b =
  ...;;

let exercise_14_test_fake_not = (test_not fake_not = true);; *)

(* ********** *)


(* Exercise 16 *)

let exercise_16 = "mandatory";;

let test_twice_less_than candidate =(candidate 100 200 300 = true)
&& (candidate 100 200 400 = true)
&& (candidate 100 400 500 = true)
&& (candidate 100 100 300 = false)
&& (candidate 200 400 400 = false)
&& (candidate 100 50 60 = false)
(* etc. *);;

let twice_less_than_v0 i j k =
  i < j && j < k;;

let exercise_16_positive_test_a = (test_twice_less_than twice_less_than_v0 =
true);;


let twice_less_than_v1 i j k =
  i = j && j = k;;

let exercise_16_negative_test_a = (test_twice_less_than twice_less_than_v1 =
false);;

let twice_less_than_v2 i j k =
  i > j && j > k;;

let exercise_16_negative_test_b = (test_twice_less_than twice_less_than_v2 =
false);;



(* Can you implement a fake twice_less_than function,
   i.e., one that is incorrect but that passes the unit tests? *)
```

```
let fake_twice_less_than i j k =
  abs(i) < abs(j) && abs(j) < abs(k);;

let exercise_16_test_fake_twice_less_than = (test_twice_less_than
fake_twice_less_than = true);;


(* ********** *)

(* Exercise 18 *)

let exercise_18 = "mandatory";;

let test_sum_odd candidate =
  (candidate 1 = (2*0+1)+(2*1+1))
&& (candidate 2 = (2*0+1)+(2*1+1)+(2*2+1))
&& (candidate 3 = (2*0+1)+(2*1+1)+(2*2+1)+(2*3+1))
&& (candidate 4 = (2*0+1)+(2*1+1)+(2*2+1)+(2*3+1)+(2*4+1))
&& (candidate 5 = (2*0+1)+(2*1+1)+(2*2+1)+(2*3+1)+(2*4+1)+(2*5+1))
(* etc. *);;


let sq_num i = (i+1)*(i+1);;

let exercise_18_positive_test = (test_sum_odd sq_num = true);;

(* ********** *)

let end_of_file = "week-03_exercises-updated.ml";;

(* remember to update the digital signature below *)

(*
        OCaml version 4.12.0

al exercise_13 : string = "mandatory"
val test_successor : (int -> int) -> bool = <fun>
val exercise_13_positive_test_a : bool = true
val exercise_13_positive_test_b : bool = true
val exercise_13_positive_test_c : bool = true
val exercise_13_negative_test_a : bool = true
val exercise_13_negative_test_b : bool = true
val exercise_13_negative_test_c : bool = true
val fake_successor : int -> int = <fun>
val exercise_13_test_fake_successor : bool = true
val exercise_14 : string = "mandatory"
val test_not : (bool -> bool) -> bool = <fun>
val exercise_14_positive_test_a : bool = true
val exercise_14_negative_test_a : bool = true
val neg_v1 : 'a -> bool = <fun>
val exercise_14_positive_test_b : bool = true
val neg_v2 : bool -> bool = <fun>
val exercise_14_positive_test_b : bool = true
val exercise_16 : string = "mandatory"
val test_twice_less_than : (int -> int -> int -> bool) -> bool = <fun>
val twice_less_than_v0 : 'a -> 'a -> 'a -> bool = <fun>
val exercise_16_positive_test_a : bool = true
val twice_less_than_v1 : 'a -> 'a -> 'a -> bool = <fun>
```

```
val exercise_16_negative_test_a : bool = true
val twice_less_than_v2 : 'a -> 'a -> 'a -> bool = <fun>
val exercise_16_negative_test_b : bool = true
val fake_twice_less_than : int -> int -> int -> bool = <fun>
val exercise_16_test_fake_twice_less_than : bool = true
val exercise_18 : string = "mandatory"
val test_sum_odd : (int -> int) -> bool = <fun>
val sq_num : int -> int = <fun>
val exercise_18_positive_test : bool = true
val end_of_file : string = "week-03_exercises-updated.ml"
*)
```