CPSC 326: Homework 7 *Due: Tuesday, Apr. 6*

Reading Assignment. Read the following sections in the textbook:

- Sections 8.1-8.2: Data Types and Type Systems
- Sectoin 11.1: Computations without State

Programming Homework. The goal of this assignment is to get started using OCaml. Note that you can download and run OCaml on your own computer (available at https://ocaml.org). OCaml is also available for use on the shared `ada` server and on the department's virtual machine. Your job is to implement the following functions *from scratch* in a file `hw7.ml` (i.e., without just calling functions provided by OCaml). In addition you must:

- only use the OCaml constructs we've discussed so far in class or as provided in the function description (if you go beyond what we've done, you'll receive no points for the question);
- follow the general style guide provided by OCaml (https://ocaml.org/learn/tutorials/guidelines.html)
- appropriately comment your code throughout including a file header with your name, file name, the date, and a brief description; and
- create sufficient test cases for your functions to ensure they work correctly.

The following ten functions must be implemented as stated above. If you have questions on how any of the following are supposed to work, please ask either during class or on piazza:

1. Implement a function `my_min x y` that evaluates to the minimum of the values `x` and `y`. For example, `my_min 2 3`, `my_min 4 2`, and `my_min 2 2` should each evaluate to `2`.
2. Implement a function `my_median x y z` that evaluates to the middle value of `x`, `y`, and `z`. For example, `my_median 1 2 3`, `my_median 3 1 2`, and `my_median 2 3 2` should each evaluate to `2`. In addition, `my_median 2 2 2` and `my_median 2 1 2` should each also evaluate to `2`.
3. Implement a function `my_triangle_area base height` that computes the triangle height given a base and a height value (as floats).
4. Implement a function `my_circle_area radius` that computes the area of a circle given the radius (as a float). Note that to raise a float value to a power you can use the exponentiation operator `**`, e.g., `v ** 3.` would cube the value `v`.
5. Implement a function `my_midpoint (x1,y1) (x2,y2)` to compute the midpoint between two points. Assume the points are given as floats. For example `my_midpoint (1.,1.) (3.,5.)` would evaluate to `(2.,3.)`. Note this function uses 2-tuples of floats to represent points.

6. Implement a function `my_manhattan_distance (x1,y1) (x2,y2)` to compute the "Manhattan" (i.e., "taxicab") distance between two points (defined as the absolute distance travelled in the x-axis plus the absolute distance travelled in the y-axis, as if you are walking city blocks to get from location 1 to location 2). Assume the points are given as floats. For example `my_manhattan_distance (1.,1.) (3.,4.)` and `my_manhattan_distance (3.,1.) (1.,4.)` would both evaluate to `5.`. To compute an absolute float value, you can use the OCaml `abs_float` function.

7. Implement a function `my_euclidean_distance (x1,y1) (x2,y2)` to compute the "Euclidean" distance (i.e., the straight-line distance from location 1 to location 2) between two points. For example, `my_euclidean_distance (1.,1.) (4.,5.)` should evaluate to `5.`. To take the square root, you can use the OCaml `sqrt` function.

8. Implement a function `my_range_sum v1 v2` that returns the sum of values starting at `v1` and ending at `v2`. For example, `my_range_sum 2 2` is `2`, `my_range_sum 2 3` is `5` (i.e., `2 + 3`), `my_range_sum 2 4` is `9` (i.e., `2 + 3 + 4`), and `my_range_sum 2 1` is `0`. You must write this as a recursive function.

9. Implement a function `my_gcd x y` that returns the greatest common divisor of two integers `x` and `y`. You must use the recursive division-based version of Euclid's Algorithm. You can use the OCaml infix function `mod` in your function.

10. Implement mutually recursive versions of even and odd functions `my_even x` and `my_odd x`. Your functions should implement the following:

```
bool even(int x) {
  if (x < 0)
    return false;
  else if (x == 0)
    return true;
  else
    return odd(x - 1);
}
bool odd(int x) {
  if (x < 0)
    return false;
  else if (x == 1)
    return true;
  else
    return even(x - 1);
}
```

For testing, we'll use our own "poor man's" unit testing approach (instead of a unit testing framework like OUnit). In particular, your program should be structured as follows:

```
(*
   Name: ...
   File: hw7.ml
   Date: Spring 2021
   Desc: ...
*)

(* Question 1: *)
let my_min x y z =
  ...
;;

(* Question 2:*)
let my_median x y z =
  ...
;;

...

(* Testing *)

let assert_equal v1 v2 msg =
  let cond = v1 = v2 in
  assert (if not cond then print_endline ("TEST FAILED: " ^ msg) ; cond)
;;

(* Question 1: my_min tests *)
assert_equal 1 (my_min 1 2) "1 = my_min 1 2";;
assert_equal 1 (my_min 2 1) "1 = my_min 2 1";;
assert_equal 1 (my_min 1 1) "1 = my_min 1 1";;

...
```

To check that your program "passes" the tests, from the command line simply run:

```
$ ocaml hw7.ml
```

Homework Submission. All homework must be submitted through GitHub Classroom. A link for each assignment will be posted on Piazza when the homework is assigned. Be sure all of your

code is pushed by the due date (which you can double check for your repository using the GitHub website). Each programming assignment is worth **35 points**. The points are allocated based on the following.

- **Correct and Complete (25 points)**. Your code must correctly and completely do the requested tasks using the requested techniques. Note that for most assignments you will be provided a *partial* set of test cases to help you determine a *minimal level* of correctness. If your program fails any of the provided test cases you will only receive partial credit. *Note that passing the given test cases does not mean your work is complete nor correct.* Your assignment will also be graded with additional test cases (not provided to you) that will help the graders determine the extent of your solution and your final score. Note that for C++ code, correctness also implies properly handling the creation and deletion of dynamic memory (i.e., the absence of memory leaks).

- **Evidence and Quality of Testing (5 points)**. As part of your homework assignments you must develop additional test cases beyond those given to you to ensure your program is correct and complete. These test cases must be turned in with your assignment. You will be graded on the scope and quality of the additional test cases you provide.

- **Formatting and Comments (5 points)**. Your code must be formatted consistently and appropriately for the language used. For C++, you must follow the provided style guide (see the course webpage).You must also comment your code and test cases, which at a minimum must include a file heading (see examples provided), function comments, and meaningfully selected variable, class, and function names. See the assignment for style guides in other languages.