MP 1 – Basic OCaml

CS 421 – Fall 2011

Revision 1.2

Assigned August 23, 2011
Due August 30, 2011, 11:59 PM
Extension 48 hours (penalty 20% of total points possible)

1 Change Log

1.2 Correct type of example output for `swap_pair`.

1.1 The input type for the function in Problem 4 should be `float`, as in the example code, not integer, as previously stated the problem text.

1.0 Initial Release.

2 Objectives and Background

The purpose of this MP is to test the student's ability to

- start up and interact with OCaml;
- define a function;
- write code that conforms to the type specified (this includes understanding simple Ocaml types, including functional ones);

Another purpose of MPs in general is to provide a framework to study for the exam. Several of the questions on the exam will appear similar to the MP problems. By the time of the exam, your goal is to be able to solve any of the following problems with pen and paper in less than 2 minutes.

3 Problems

Note: In the problems below, you do not have to begin your definitions in a manner identical to the sample code, which is present solely for guiding you better. However, you have to use the indicated name for your functions, and the functions will have to conform to any type information supplied, and have to yield the same results as any sample executions given.

1. (1 pt) Declare a variable `x` with the value `32.7`. It should have type `float`.

2. (1 pt) Declare a variable `salute` with a value of `"Greetings, my friend!"`. It should have the type of `string`.

3. (2 pts) Write a function `times_13` that returns the result of mulitplying a given integer by 13.

```
# let times_13 n = ... ;;
val times_13 : int -> int = <fun>
# times_13 7;;
- : int = 91
```

4. (2 pts) Write a function `square_plus_x y` that returns the result of adding the value of `x` from Problem 1 to the square of the float-valued input.

```
# let square_plus_x y = ... ;;
val square_plus_x : float -> float = <fun>
# square_plus_x 23.17;;
- : float = 569.548900000000117
```

5. (3 pts) Write a function `hail` that takes a string, which is assumed to be a person's name, and prints out a greeting as follows: If the name is `"Elsa"`, it prints out the string

```
"Wayell, hah theya, Ayelsa!"
```

(no "newline" at the end, and it does not print the quotation marks). For any other string, it first prints out `"Dear, "`, followed by the given name, followed by `". I wish you the best in CS421."`, followed by a "newline".

```
# let hail name = ... ;
val hail : string -> unit = <fun>
# hail "Thomas";;
Dear, Thomas. I wish you the best in CS421.
- : unit = ()
```

6. (4 pts) Write a function `has_smallest_abs` that, when given one integer and then another, returns the one that has the smallest absolute value. If they have the same absolute value, it should return the second value given.

```
# let has_smallest_abs m n = ... ;;
val has_smallest_abs : int -> int -> int = <fun>
# has_smallest_abs 4 6;;
- : int = 4
```

7. (3 pts) Write a function `swap_pair` that takes a pair and returns the pair that has the components in the opposite order.

```
# let swap_pair (x,y) = ... ;;
val swap_pair : 'a * 'b -> 'b * 'a = <fun>
# swap_pair (3, 5);;
- : int * int = (5, 3)
```

8. (5 pts) Write a function `pair_app` that takes a pair of functions as a first argument and then takes a second argument and returns the pair resulting from applying the first function to the second argument paired with the second function applied to the second argument.

```
# let pair_app (f,g) x = ... ;;
val pair_app : ('a -> 'b) * ('a -> 'c) -> 'a -> 'b * 'c = <fun>
# pair_app (times_13, fun a -> a + 2) 6;;
- : int * int = (78, 8)
```