MP 0 – Basic OCaml

CS 421 – Spring 2011
Revision 1.0

Assigned January 18, 2011
Due January 19, 2011, 23:59
Extension 48 hours (penalty 20% of total points possible)

1 Change Log

1.0 Initial Release.

2 Objectives and Background

The purpose of this MP is to get you started using OCaml, which we will both study and use extensively in this class. OCaml is an example of a "functional language." In this MP, you will access the EWS system, download the package of code needed to do this MP, write solutions to simple problems, test them, and submit them using the handin system. (You may also download OCaml onto your own computers, to be less reliant on the EWS machines, although you will always need to use the EWS machines to run handin.)

This MP is ungraded, but it is strongly recommended. Problems are given point values below just to make the assignment more "realistic." We don't believe the assignment should take more than two hours at the most.

Note that you will be doing this MP prior to having any instruction on OCaml in class. The assignment asks you to do only the simplest things, but obviously you need something to get you started. These resources are provided on the course website:

- The document "Basic guide to OCaml" is linked from the page where you got this MP. It contains everything you need to know about OCaml for this MP.
- The MPs page on the website contains a section (also linked from this page) entitled "Guide for Doing MPs." We wish this were less complicated, but it's the best we can do; this complexity is one reason we are assigning an MP0.
- The "Resources" and "faq" tabs on the website explain access to the EWS machines; hopefully, this is not new to you.
- Also on the Resources page are links to OCaml information; in particular, there is a link to www.ocaml.org, from which you can download OCaml.

3 Collaboration

Collaboration is NOT allowed in this assignment.

Each student needs to be familiar with our handin system, and know how to run a program in OCaml. If you are stuck on this assignment, ask for help from TAs right away.

4 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 to yield the same results as any sample executions given.

1. (1 pt) Declare a variable `a` with the value `17`. It should have type `int`.

2. (1 pt) Declare a variable `s` with the value `"Hi there"`. It should have the type of `string`.

3. (2 pts) Write a function `add_a` that adds the value of `a` from Problem 1 to its argument.

```
# let add_a n = ... ;;
val add_a : int -> int = <fun>
# add_a 13;;
- : int = 30
```

4. (2 pts) Write a function `s_paired_with_a_times` that returns the pair of the string `s` from Problem 2 paired with the result of multiplying the value `a` from Problem 1 by the integer input.

```
# let s_paired_with_a_times b = ... ;;
val s_paired_with_a_times : int -> string * int = <fun>
# s_paired_with_a_times 12;;
- : string * int = ("Hi there", 204)
```

5. (3 pts) Write a function `abs_diff` that takes two arguments of type `float` and returns the absolute value of the difference of the two. Pay careful attention to the type of this problem.

```
# let abs_diff x y = ... ;;
val abs_diff : float -> float -> float = <fun>
# abs_diff 15.0 11.5;;
- : float = 3.5
```

6. (3 pts) Write a function `greetings` that takes a string, which is assumed to be a person's name, and prints out a greeting as follows: If the name is `"Sam"`, it prints out the string

   `"Hi Sam!"`

   with no newline at the end. For any other argument, it prints out `"Hello, "`, followed by the given name, followed by `". I hope you enjoy CS421."` , followed by a newline.

```
# let greetings name = ... ;;
val greetings : string -> unit = <fun>
# greetings "Angela";;
Hello, Angela. I hope you enjoy CS421.
- : unit = ()
```

7. (3 pts) Write a function `greetstring` that is similar to the previous problem, but instead of *printing* the specified string, it *returns* the string as its result.

```
# let greetstring name = ... ;;
val greetings : string -> string = <fun>
# greetstring "Angela";;
- : string = "Hello, Angela. I hope you enjoy CS421."
```

8. (3 pts) Write a function `sign` that, when given an integer, returns `1` if the integer is positive, `0` if the integer is zero and `-1` if the integer is negative.

```
# let sign n = ... ;;
val sign : int -> int = <fun>
# sign 4;;
- : int = 1
```