CISS445: Programming Languages
Assignment 2

OBJECTIVES: This is the first of several OCAML assignments where the main object is to help you learn the basic OCAML language, including basic types and operators, tuples and lists with their operations, declarations, recursion, and matchings.

This is an easy assignment. Therefore no discussion is allowed.

Furthermore you MUST NOT use OCAML libraries. For instance if a question ask you to implement the `head` function, you CANNOT use the `List.hd` function which achieves the same thing. Your work will be automatically graded by a program that will catch such library usage. Any assignment using disallowed libraries will get an immediate 0.

How to Work With OCAML Files

Although the OCAML interpreter is useful for experiments (because of the immediate feedback), it's not convenient because of the limited features of the editor within the interpreter.

There are many ways to develop and run/test ocaml programs productively. The easiest is the following and is enough for completing most of the assignments.

First use text editor and write the following test.ml and save it:

```
(*
    Name: smaug
    File: test.ml
*)

print_string "hello world\n";;
let f = fun x -> x + 1;;
print_int (f 42);;
```

Now in your linux shell, type this:

```
ocaml test.ml
```

and you will see the following output:

```
hello world
43
```

You can now toggle between your favorite text editor and your linux shell as you write and test your code.

SUBMISSION

There are several things you MUST do before you turn in your work.

1. Make a directory `a02/`. For assignment 2 question 1, create a directory `a02q01/` in `a02/`. Your solution to assignment 2 question 1 must be kept in `a02q01.ml` in directory `a02/a02q01/`. `a02q01.ml` must begin with the following preamble:

```
(*
    File: smaug
    Name: a02q01.ml
*)

exception IgnoreCase;;
exception NotImplemented;;

(* your code is below ... *)
```

with your name replacing `smaug` of course.

2. Email your files as `a02.tar.gz` to `yliow.submit@gmail` from your college email account. The subject line must be "`ciss445 a02`". To create the `a02.tar.gz` file, go to your `a02/` directory and execute these command

```
tar -cvf a02.tar .
gzip a02.tar
```

This will collect all the directories (with their files) in your `a02/` direction and put them into `a02.tar.gz`. If you wish, you can view the contents of `a02.tar.gz` by doing this:

```
tar -ztvf a02.tar.gz
```

After emailing `a02.tar.gz`, you can of course remove the file:

```
rm a02.tar.gz
```

3. You MUST make sure your OCAML file does not generate any errors or warnings. Expressions generating warnings and errors and your debug/test printing must be removed, commented out, or modified (see 4 and 5 below). You can easily test for warning and error messages when you do this at your linux shell (suppose that `$` is your linux prompt):

```
$ ocaml a02q01.ml
```

If there are no errors or warnings, then nothing is printed from OCAML and you get your linux prompt immediately. If you get something like this:

```
$ ocaml a02q01.ml
File "a02q01.ml", line 3, characters 0-3:
Unbound value foo

$
```

or something like this:

```
$ ocaml a02q01.ml
File "a02q01.ml", line 3, characters 0-1:
Syntax error

$
```

or something like this:

```
$ ocaml a02q01.ml
File "a02q01.ml", line 3, characters 11-33:
Warning: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
_::_

$
```

4. For every function that you cannot finish implementing, you must do the following. Suppose you are not done with `max` (see Q1). Note that it has three parameters. You MUST add this to your `a02q01.ml`:

```
let max x y z = raise NotImplemented;;
```

or

```
let max = fun x -> fun y -> fun z -> raise NotImplemented;;
```

5. This part applies to questions that requires pattern matching. (This does not apply to a02 since there is no pattern matching in a02.) The following is what you MUST do if you have unmatched cases in your function. Suppose you have a function f like this:

```
let f x = match x with
  0 -> 0
| 1 -> 2
| 2 -> 4
;;
```

you will get a warning for an unmatched case. You must add the follow:

```
let f x = match x with
  0 -> 0
| 1 -> 2
| 2 -> 4;;
| _ -> raise IgnoreCase
;;
```

You must ensure that the expressions in your file do not generate warnings. Of course if you do NOT have unmatched case warning, do NOT add the above or you WILL get a warning!

Q1. Write a (curried) function max that returns the largest value among three values passed in.

The following are some test cases:

```
Tests:                  Expected value
max 1 2 3;;             3
max 1 3 2;;             3
max 2 1 3;;             3
max 2 3 1;;             3
max 3 1 2;;             3
max (-3) (-2) (-1);;    -1
```

Just like in the set of integers, you have operations (i.e., addition, subtraction, etc.), you also have operations on the set of functions; assume that these functions have one float input and the compute float values. The following questions essentially defining operations on functions. Most of the solutions are the same. Therefore the key is to make sure that your function for Q2 is done correctly.

Q2. Write a function `add_func` that returns a function that is the sum of two functions.

Here are some test cases:

```
Tests:                  Expected value
let f x = x +. 1.0;;
let g x = x -. 1.0;;
let h x = x *. x;;
(add_func f g) 0.0;;    0.
(add_func f g) 1.0;;    2.
(add_func g f) 2.0;;    4.
(add_func g f) 3.0;;    6.
(add_func f h) 0.0;;    1.
(add_func f h) 1.0;;    3.
(add_func h f) 2.0;;    7.
```

Q3. Write a function `sub_func` that returns a function that is the difference of two functions.

Here are some test cases:

```
Tests:                      Expected value
let f x = x +. 1.0;;
let g x = x -. 1.0;;
let h x = x *. x;;
(sub_func f g) 0.0;;        2.
(sub_func f g) 1.0;;        2.
(sub_func g f) 2.0;;        -2.
(sub_func g f) 3.0;;        -2.
(sub_func f h) 0.0;;        1.
(sub_func f h) 1.0;;        1.
(sub_func h f) 2.0;;        1.
```

Q4. Write a function `mult_func` that returns a function that is the product of two functions.

Here are some test cases:

Tests:	Expected value
`let f x = x +. 1;;`	
`let g x = x -. 1;;`	
`let h x = x *. x;;`	
`(mult_func f g) 0.0;;`	`-1.`
`(mult_func f g) 1.0;;`	`0.`
`(mult_func g f) 2.0;;`	`3.`
`(mult_func g f) 3.0;;`	`8.`
`(mult_func f h) 0.0;;`	`0.`
`(mult_func f h) 1.0;;`	`2.`
`(mult_func h f) 2.0;;`	`12.`

Q5. Write a function `div_func` that returns a function that is the quotient two functions.

```
Tests:                              Expected value
let f x = x +. 1.0;;
let g x = x -. 1.0;;
let h x = x *. x;;
(div_func f g 0.0);;                -1.
(div_func g f 2.0);;                0.333333333333
(div_func g f 3.0);;                0.5
(div_func f h 1.0);;                2.
(div_func h f 2.0);;                1.33333333333
```

Q6. Write a function `comp_func` that returns a function that is the composition two functions. Given two functions $f(x)$ and $g(x)$, the composition of $f(x)$ and $g(x)$ computes $f(g(x))$ for a given input of x.

Here are some test cases:

```
Tests:                             Expected value
let f x = x +. 1.0;;
let g x = x -. 1.0;;
let h x = x *. x;;
(comp_func f g) 0.0;;              0.
(comp_func f g) 1.0;;              1.
(comp_func g f) 2.0;;              2.
(comp_func g f) 3.0;;              3.
(comp_func f h) 0.0;;              1.
(comp_func f h) 1.0;;              2.
(comp_func h f) 2.0;;              9.
```

Q7. Write a function `max_func` that returns the maximum of the value of two functions.

Here are some test cases:

```
Tests:                          Expected value
let f x = x +. 1.0;;
let g x = x {. 1.0;;
let h x = x *. x;;
(max_func f g 0.0);;            1.
(max_func f g 1.0);;            2.
(max_func g f 2.0);;            3.
(max_func g f 3.0);;            4.
(max_func f h 0.0);;            1.
(max_func f h 1.0);;            2.
(max_func h f 2.0);;            4.
```

Q8. First, let me describe the function. Given a function $f(x)$ we can define another

$$\frac{f(x+h)-f(x)}{h}$$

This is a function of two variables, x and h. We can write this function as $(Df)(x,h)$. For instance if $f(x)=x+3$, then $(Df)(x,h)$ is

$$\frac{(x+3+h)-(x+3)}{h}$$

and therefore $(Df)(x,h)=1$. Here's another example: Suppose $f(x)=x^2$. Then $(Df)(x,h)$ is

$$\frac{(x+h)^2-x^2}{h}$$

which gives (after simplification) $(Df)(x,h)=2x+h$. Therefore $(Df)(1.0,0.1)$ is 2.1. You can therefore think of a function D that takes a function $f(x)$ and creates another function (Df) of two variables, i.e. $(Df)(x,h)$. The goal is to write the function D. Now for the OCAML function that you need to write ...

Write an OCAML function `d` (use `d` and not `D`) that behaves as above except that the argument `h` comes first. The following are two test cases. The following expression

```
d (fun x -> x +.  3.0) 2.0 5.0
```

(the `h` value is `2.0` and the `x` value is `5.0`) gives `1.0` and the expression

```
d (fun x -> x *.  x) 0.001 1.0
```

gives `2.001`.