CS251 Programming Languages **Handout # 6**
Prof. Lyn Turbak **January 29, 2008**
Wellesley College

OCAML Exercises

OCAML is very similar to the SML programming language we used in CS235. Most of the differences are in surface syntax. See

`http://www.mpi-sws.mpg.de/~rossberg/sml-vs-ocaml.html`

for a side-by-side comparison of SML and OCAML. (This is linked from the *CS251 Resources* page.)

This handout contains lots of simple OCAML expressions that you should try out interactively in an OCAML interpreter as part of learning the language. Predict what the answer to each expression will be, and then try it out. The interactive nature of OCAML makes it easy to experiment!

```
# 1 + 2;;
```

```
# 1 + 2
;;
```

```
# let a = 3 + 4;;
```

```
# let a=3+4;;
```

```
# a * a;;
```

```
# let dbl = fun x -> x * 2;;
```

```
# dbl 10;;
```

```
# dbl (10);;
```

```
# (dbl 10);;
```

```
# dbl (dbl 10);;

# dbl dbl 10;;

# (fun x -> x + 1) 10;;

# let inc x = x + 1;;

# inc 10;;

# let app5 f = f 5;;

# app5 inc;;

# app5 dbl;;

# let b = a * 2;;

# let adda x = x + a;;

# adda 10;;

# let a = 42;;

# b;;
```

```
# adda 10;;


# let f n = if n > 10 then 2 * n else n * n;;


# f 20;;


# f 5;;


# let rec fact n =
    if n = 0 then
      1
    else
      n * (fact (n-1));;


# fact 5;;


# fact 46;;


# fact 47;;


# let fact2 n =
    if n = 0 then
      1
    else
      n * (fact2 (n-1));;


# 1 = 1;;
```

```
# 2 = 3;;
```

```
# "foobar";;
```

```
# String.length "foobar";;
```

```
# String.get "foobar" 5;;
```

```
# "baz" ^ "quux" ^ (string_of_int 17);;
```

```
# (2 * 3, 4 < 5, "foo" ^ "bar", String.get "baz" 2);;
```

```
# let swap (a,b) = (b,a);;
```

```
# swap (1+2,3=4);;
```

```
# swap(swap(1+2,3=4));;
```

```
# let step (a,b) = (a + b, a*b);;
```

```
# step (1,2);;
```

```
# step (step (1,2));;
```

```
# let (x,y) = step (step (1,2)) in x+y;;
```

```
# let rec stepuntil ((a,b),limit) =
    if a >= limit then
      (a,b)
    else
      stepuntil(step(a,b),limit);;



# stepuntil ((1,2), 100);;



# let print_pair (a,b) =
    print_string ("(" ^ (string_of_int a) ^ ","
                  ^ (string_of_int b) ^ ")\n");;



# let rec stepuntil ((a,b),limit) =
   if a >= limit then
     (a,b)
   else
    (print_pair (a,b);
     stepuntil(step(a,b),limit));;



# stepuntil ((1,2),100);;



# let sumDivisors n =
    if n <= 0 then
      0
    else
      let rec sum d =
        if d == 0 then
          0
        else if (n mod d) == 0 then
          d + sum (d-1)
        else
          sum (d-1)
      in sum (n-1);;


# sumDivisors 12;;
```

```
# let numDivisors n =
    if n <= 0 then
      0
    else
      let rec sum d =
        if d == 0 then
          0
        else if (n mod d) == 0 then
          1 + sum (d-1)
        else
          sum (d-1)
      in sum (n-1);;



# numDivisors 12;;



# let numAndSumDivisors n =
    if n <= 0 then
     (0,0)
    else
      let rec sum d =
        if d == 0 then
          (0,0)
        else if (n mod d) == 0 then
          let (n,s) = sum (d-1)
            in (1+n,d+s)
        else
          sum (d-1)
      in sum (n-1);;



# numAndSumDivisors 12;;



# let avg1 (a,b) = (a+b)/2;;



# avg1 (10,20);;



# let avg2 a b = (a+b)/2;;
```

```
# avg2 10 20;;


# app5 (avg2 15);;


# app5 (fun x -> avg1(15,x));;
```

Rather than typing in declarations interactively, we can instead put them in a file. For example, we can collect all the declarations from above into a file named `FunTest.ml` (Fig. 1). We can then load them into OCAML via the `#use` directive:

```
# #use "FunTest.ml";;
val a : int = 7
val dbl : int -> int = <fun>
val inc : int -> int = <fun>
val app5 : (int -> 'a) -> 'a = <fun>
val b : int = 14
val adda : int -> int = <fun>
val a : int = 42
val f : int -> int = <fun>
val fact : int -> int = <fun>
val swap : 'a * 'b -> 'b * 'a = <fun>
val step : int * int -> int * int = <fun>
val stepuntil : (int * int) * int -> int * int = <fun>
val print_pair : int * int -> unit = <fun>
val stepuntil : (int * int) * int -> int * int = <fun>
val sumDivisors : int -> int = <fun>
val numDivisors : int -> int = <fun>
val numAndSumDivisors : int -> int * int = <fun>
val avg1 : int * int -> int = <fun>
val avg2 : int -> int -> int = <fun>
```

```
let a = 3 + 4;;

let dbl = fun x -> x * 2;;

let inc x = x + 1;;

let app5 f = f 5;;

let b = a * 2;;

let adda x = x + a;;

let a = 42;;

let f n = if n > 10
          then 2 * n
          else n * n;;

let rec fact n =
  if n = 0 then
    1
  else
    n * (fact (n-1));;

let swap (a,b) = (b,a);;

let step (a,b) = (a + b, a*b);;

let rec stepuntil ((a,b),limit) =
  if a >= limit then
    (a,b)
  else
    stepuntil(step(a,b),limit);;

let print_pair (a,b) =
  print_string
   ("("
     ^ (string_of_int a)
     ^ ","
     ^ (string_of_int b)
     ^ ")\n");;

let rec stepuntil ((a,b),limit) =
  if a >= limit then
    (a,b)
  else
    (print_pair (a,b);
     stepuntil(step(a,b),limit));;
```

```
let sumDivisors n =
  if n <= 0 then
    0
  else
    let rec sum d =
      if d == 0 then
        0
      else if (n mod d) == 0 then
        d + sum (d-1)
      else
        sum (d-1)
    in sum (n-1);;

let numDivisors n =
  if n <= 0 then
    0
  else
    let rec sum d =
      if d == 0 then
        0
      else if (n mod d) == 0 then
        1 + sum (d-1)
      else
        sum (d-1)
    in sum (n-1);;

let numAndSumDivisors n =
  if n <= 0 then
    (0,0)
  else
    let rec sum d =
      if d == 0 then
        (0,0)
      else if (n mod d) == 0 then
        let (n,s) = sum (d-1)
        in (1+n,d+s)
      else
        sum (d-1)
    in sum (n-1);;

let avg1 (a,b) = (a+b)/2;;

let avg2 a b = (a+b)/2;;
```

Figure 1: Contents of the file `FunTest.ml`.