CS450: Progrmg Languages & Compilers
Fall 2014

Homework 2

University of Illinois, Urbana-Champaign,

Due: June 13, 2015, 5.00 pm. Submit PDF + code files on Moodle.

1 Theoretical Questions (40 points)

Problems

Problem 1: (24 points)

Below is a fragment of OCaml code, with various program points indicated by numbers with comments.

```
let p1 = (1., 3.);;
let p2 = (2., 5.);;
(* 1 *)
let slope (x,y) = y /. x;;
(* 2 *)
let sub (x1,y1) (x2,y2) = (x2 - x1, y2 - y1);;
(* 3 *)
let slope p1 p2 = slope (sub p1 p2);;
(* 4 *)
let slope_p2 = slope p2;;
(* 5 *)
let p2 = (3., 9.);;
(* 6 *)
slope_p2 p1;;
(* 7 *)
slope p1 p2;;
(* 8 *)
```

- (12 points) For each of program points 1, 2, 3, 4, 5 and 6, please describe the environment in effect after evaluation has reached that point. You may assume that the evaluation begins in an empty environment, and that the environment is cumulative thereafter. The program points are supposed to indicate points at which all complete preceding declarations (including local ones) have been fully evaluated. In describing the environments 1 through 5, you may use set notation, as done in class, or you may use the update operator +. If you use set notation, no duplicate bindings should occur. The answer for program point 6 should be written out fully in set notation without the update operator or duplications.

  Solution:

  1. $\rho_1 = \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\}$
  2. $\rho_2 = \{\texttt{slope} \to \langle (\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \rho_1 \rangle\} + \rho_1$,
  3. $\rho_3 = \{\texttt{sub} \to \langle (\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun}\ (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_2 \rangle\} + \rho_2$
  4. $\rho_4 = \{\texttt{slope} \to \langle \texttt{p1} \mapsto \texttt{fun p2} \mapsto \texttt{slope}\ (\texttt{sub p1 p2}), \rho_3 \rangle\} + \rho_3$
  5. $\rho_5 = \{\texttt{slope\_p2} \to \langle \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub p1 p2}), \{\texttt{p1} \to (2., 5.)\} + \rho_3 \rangle\} + \rho_4$

6. $\rho_6 = \{\texttt{p2} \to (3., 9.),$
$\texttt{slope\_p2} \to \langle \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \rho_{\texttt{slope\_p2}} \rangle,$
$\texttt{slope} \to \langle \texttt{p1} \mapsto \texttt{fun}\ \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \rho_{\texttt{slope}} \rangle$
$\texttt{sub} \to \langle (\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun}\ (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{sub}} \rangle$
$\texttt{p1} \to (1., 3.)\}$

where
$\rho_{slope\_p2} = \{\ \texttt{p1} \to (2., 5.),$
$\texttt{sub} \to \langle (\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun}\ (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{sub}} \rangle,$
$\texttt{slope} \to \langle (\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\} \rangle,$
$\texttt{p2} \to (2., 5.)\},$
$\rho_{slope} = \{\texttt{sub} \to \langle (\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun}\ (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{sub}} \rangle,$
$\texttt{slope} \to \langle (\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\} \rangle,$
$\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\},$
$\rho_{sub} = \{\texttt{slope} \to \langle (\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\} \rangle,$
$\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\}.$

- (4 points) Show step by step how (* 5 *) is evaluated.

Solution:

$\texttt{Eval}(\texttt{slope}\ \texttt{p2}, \rho_4)$

$$\begin{aligned} &= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_4), \texttt{Eval}(\texttt{p2}, \rho_4)), \rho_4) \\ &= \texttt{Eval}(\texttt{app}(\langle \texttt{p1} \mapsto \texttt{fun}\ \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \rho_3 \rangle, (2., 5.)), \rho_4) \\ &= \texttt{Eval}(\texttt{fun}\ \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \{\texttt{p1} \to (2., 5.)\} + \rho_3) \\ &= \langle \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \{\texttt{p1} \to (2., 5.)\} + \rho_3 \rangle \end{aligned}$$

- (4 points) Show step by step how `slope_p2 p1` is evaluated.

Solution:

$\texttt{Eval}(\texttt{slope\_p2}\ \texttt{p1}, \rho_6)$

$$\begin{aligned} &= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope\_p2}, \rho_6), \texttt{Eval}(\texttt{p1}, \rho_6)), \rho_6) \\ &= \texttt{Eval}(\texttt{app}(\langle \texttt{p2} \mapsto \texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \rho_{\texttt{slope\_p2}} \rangle, (1., 3.)), \rho_6) \\ &= \texttt{Eval}(\texttt{slope}\ (\texttt{sub}\ \texttt{p1}\ \texttt{p2}), \rho_{\texttt{t1}} = \{\texttt{p2} \to (1., 3.)\} + \rho_{\texttt{slope\_p2}}) \\ &= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_{\texttt{t1}}), \texttt{Eval}(\texttt{sub}\ \texttt{p1}\ \texttt{p2}, \rho_{\texttt{t1}})), \rho_{\texttt{t1}}) \end{aligned}$$

*** begin Subcomputation 1 ***

$\texttt{Eval}(\texttt{sub p1 p2}, \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{sub p1}, \rho_{\texttt{t1}}), \texttt{Eval}(\texttt{p2}, \rho_{\texttt{t1}})), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{sub}, \rho_{\texttt{t1}}), \texttt{Eval}(\texttt{p1}, \texttt{rho}_{\texttt{t1}})), \rho_{\texttt{t1}}), (1., 3.)), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\langle(\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun } (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{sub}}\rangle, (2., 5.)), \rho_{\texttt{t1}}), (1., 3.)), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{fun } (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t2}} = \{\texttt{x1} \to 2., \texttt{y1} \to 5.\} + \rho_{\texttt{sub}}), (1., 3.)), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\langle(\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t2}}\rangle, (1., 3.)), \rho_{\texttt{t1}})$

$= \texttt{Eval}((\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t3}} = \{\texttt{x2} \to 1., \texttt{y2} \to 3.\} + \rho_{\texttt{t2}})$

$= \texttt{Eval}((\texttt{Eval}(\texttt{x2} - .\texttt{x1}, \rho_{\texttt{t3}}), \texttt{Eval}(\texttt{y2} - .\texttt{y1}, \rho_{\texttt{t3}})), \rho_{\texttt{t3}})$

$= \texttt{Eval}((\texttt{Eval}(\texttt{Eval}(\texttt{x2}, \rho_{\texttt{t3}}) - .\texttt{Eval}(\texttt{x1}, \rho_{\texttt{t3}}), \rho_{\texttt{t3}}), \texttt{Eval}(\texttt{Eval}(\texttt{y2}, \rho_{\texttt{t3}}) - .\texttt{Eval}(\texttt{y1}, \rho_{\texttt{t3}}), \rho_{\texttt{t3}})), \rho_{\texttt{t3}})$

$= \texttt{Eval}((\texttt{Eval}(1. - .2., \rho_{\texttt{t3}}), \texttt{Eval}(3. - .5., \rho_{\texttt{t3}})), \rho_{\texttt{t3}})$

$= \texttt{Eval}((-1., -2.), \rho_{\texttt{t3}})$

$= (-1., -2.)$

*** end Subcomputation 1 ***

$\texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_{\texttt{t1}}), \texttt{Eval}(\texttt{sub p1 p2}, \rho_{\texttt{t1}})), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{app}(\langle(\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\}\rangle, (-1., -2.)), \rho_{\texttt{t1}})$

$= \texttt{Eval}(\texttt{y}/.\texttt{x}, \rho_{\texttt{t4}} = \{\texttt{x} \to -1., \texttt{y} \to -2., \texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\})$

$= \texttt{Eval}(\texttt{Eval}(\texttt{y}, \rho_{\texttt{t4}})/.\texttt{Eval}(\texttt{x}, \rho_{\texttt{t4}}), \rho_{\texttt{t4}})$

$= \texttt{Eval}(-2./. - 1., \rho_{\texttt{t4}})$

$= 2.$

- (4 points) Show step by step how `slope p1 p2` is evaluated.

Solution:

$\texttt{Eval}(\texttt{slope p1 p2}, \rho_6)$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope p1}, \rho_6), \texttt{Eval}(\texttt{p2}, \rho_6)), \rho_6)$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_6), \texttt{Eval}(\texttt{p1}, \rho_6)), \rho_6), (3., 9.)), \rho_6)$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\langle\texttt{p1} \mapsto \texttt{fun p2} \mapsto \texttt{slope } (\texttt{sub p1 p2}), \rho_{\texttt{slope}}\rangle, (1., 3.)), \rho_6), (3., 9.)), \rho_6)$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{fun p2} \mapsto \texttt{slope } (\texttt{sub p1 p2}), \rho_{\texttt{t5}} = \{\texttt{p1} \to (1., 3.)\} + \rho_{\texttt{slope}}), (3., 9.)), \rho_6)$

$= \texttt{Eval}(\texttt{app}(\langle\texttt{p2} \mapsto \texttt{slope } (\texttt{sub p1 p2}), \rho_{\texttt{t5}}\rangle, (3., 9.)), \rho_6)$

$= \texttt{Eval}(\texttt{slope } (\texttt{sub p1 p2}), \rho_{\texttt{t6}} = \{\texttt{p2} \to (3., 9.)\} + \rho_{\texttt{t5}})$

$= \texttt{Eval}(\texttt{slope } (\texttt{sub p1 p2}), \rho_{\texttt{t6}})$

$= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_{\texttt{t6}}), \texttt{Eval}(\texttt{sub p1 p2}, \rho_{\texttt{t6}})), \rho_{\texttt{t6}})$

*** begin Subcomputation 1 ***

$\texttt{Eval}(\texttt{sub p1 p2}, \rho_{\texttt{t6}})$

$$
\begin{aligned}
&= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{sub p1}, \rho_{\texttt{t6}}), \texttt{Eval}(\texttt{p2}, \rho_{\texttt{t6}})), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{sub}, \rho_{\texttt{t6}}), \texttt{Eval}(\texttt{p1}, \texttt{rho}_{\texttt{t6}})), \rho_{\texttt{t6}}), (3., 9.)), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{app}(\langle(\texttt{x1}, \texttt{y1}) \mapsto \texttt{fun } (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{sub}}\rangle, (1., 3.)), \\
&\qquad \rho_{\texttt{t6}}), (3., 9.)), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{fun } (\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t7}} = \{\texttt{x1} \to 1., \texttt{y1} \to 3.\} + \rho_{\texttt{sub}}), \\
&\qquad (3., 9.)), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}(\texttt{app}(\langle(\texttt{x2}, \texttt{y2}) \mapsto (\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t7}}\rangle, (3., 9.)), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}((\texttt{x2} - .\texttt{x1}, \texttt{y2} - .\texttt{y1}), \rho_{\texttt{t8}} = \{\texttt{x2} \to 3., \texttt{y2} \to 9.\} + \rho_{\texttt{t7}}) \\
&= \texttt{Eval}((\texttt{Eval}(\texttt{x2} - .\texttt{x1}, \rho_{\texttt{t8}}), \texttt{Eval}(\texttt{y2} - .\texttt{y1}, \rho_{\texttt{t8}})), \rho_{\texttt{t8}}) \\
&= \texttt{Eval}((\texttt{Eval}(\texttt{Eval}(\texttt{x2}, \rho_{\texttt{t8}}) - .\texttt{Eval}(\texttt{x1}, \rho_{\texttt{t8}}), \rho_{\texttt{t8}}), \\
&\qquad \texttt{Eval}(\texttt{Eval}(\texttt{y2}, \rho_{\texttt{t8}}) - .\texttt{Eval}(\texttt{y1}, \rho_{\texttt{t8}}), \rho_{\texttt{t8}})), \rho_{\texttt{t8}}) \\
&= \texttt{Eval}((\texttt{Eval}(3. - .1., \rho_{\texttt{t8}}), \texttt{Eval}(9. - .3., \rho_{\texttt{t8}})), \rho_{\texttt{t8}}) \\
&= \texttt{Eval}((2., 6.), \rho_{\texttt{t8}}) \\
&= (2., 6.)
\end{aligned}
$$

*** end Subcomputation 1 ***

$\texttt{Eval}(\texttt{app}(\texttt{Eval}(\texttt{slope}, \rho_{\texttt{t6}}), \texttt{Eval}(\texttt{sub p1 p2}, \rho_{\texttt{t6}})), \rho_{\texttt{t6}})$

$$
\begin{aligned}
&= \texttt{Eval}(\texttt{app}(\langle(\texttt{x}, \texttt{y}) \mapsto \texttt{y}/.\texttt{x}, \{\texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\}\rangle, (2., 6.)), \rho_{\texttt{t6}}) \\
&= \texttt{Eval}(\texttt{y}/.\texttt{x}, \rho_{\texttt{t9}} = \{\texttt{x} \to 2., \texttt{y} \to 6., \texttt{p1} \to (1., 3.), \texttt{p2} \to (2., 5.)\}) \\
&= \texttt{Eval}(\texttt{Eval}(\texttt{y}, \rho_{\texttt{t9}})/.\texttt{Eval}(\texttt{x}, \rho_{\texttt{t9}}), \rho_{\texttt{t9}}) \\
&= \texttt{Eval}(6./.2., \rho_{\texttt{t9}}) \\
&= 3.
\end{aligned}
$$

Problem 2: (16 points)

Below is a fragment of OCaml code.

```
let f g x =
  (let r =
    if ((print_string "a"; x > 5) && (g(); x > 10))
    then
       (print_string "b"; x - 7)
    else
       let z = (print_string "c"; 15) in (print_string "d"; z)
    in (g(); r));;
let u = (f (fun () -> print_string "e\n") (f (fun () -> print_string "f\n") 3));;
```

Describe everything that is displayed on the screen after this code is pasted into an interactive OCaml session. This should include both the type information that the compiler gives back for each declaration, and any other things printed to the screen. Give explanations for all the other things printed, and the order in which they are printed. For the type information, no explanation is required but it should be correct.

Solution:

The given code in OCaml gives the following output.

```
# let f g x =
(let r =
if ((print_string "a"; x > 5) && (g(); x > 10))
then
(print_string "b"; x - 7)
else
let z = (print_string "c"; 15) in (print_string "d"; z)
in (g(); r));;
val f : (unit -> 'a) -> int -> int = <fun>
# let u = (f (fun () -> print_string "e\n") (f (fun () -> print_string "f\n") 3));;
acdf
ae
be
val u : int = 8
```

1. The expression `(f (fun () -> print_string "e\n") (f (fun () -> print_string "f\n") 3))` is evaluated and the value is bound to `u`. The following is output by ocaml:

   ```
   acdf
   ae
   be
   val u : int = 8
   ```

2. This involves invoking function `f` with two arguments: `(fun () -> print_string "e\n")` and `(f (fun () -> print_string "f\n") 3)`

3. The right-most argument gets evaluated first. Thus, function `f` is invoked again with two arguments: `(fun () -> print_string "f\n")` and `3`

4. The invocation of function `f` mentioned in step 3 results in the following:

   (a) the argument `(fun () -> print_string "f\n")` binds to the formal parameter `g` and the argument `3` binds to the formal parameter `x`,

   (b) in the body of `f`, the value of the `if-then-else` expression is evaluated and bound to `r` as follows:

      i. the expression `(print_string "a"; x > 5) && (g(); x > 10)` in the `if` condition is first evaluated. The two arguments of the `&&` operator in this expression are `(print_string "a"; x > 5)` and `(g(); x > 10)`. Since `&&` uses short-circuit evaluation, the left argument is evaluated first, which causes the string `"a"` to be output, but evaluates to `false` since `3 < 5`. Since the left argument of the `&&` operator is found to be `false`, owing to short-circuit evaluation, the right argument is not evaluated anymore and the `if` condition is evaluated to be `false`.

      ii. the expression under the `else` part is then evaluated. First, the expression being bound to `z` in the `let` expression is evaluated. This causes the string `"c"` to be output and the value `15` to be bound to `z`. Next, the main body of the `let` expression is evaluated, and this causes the string `"d"` to be output and the value of the `else` part evaluates to the value of the expression `z`, which is `15`.

iii. the value of the expression under the `else` part is bound to `r`. Thus, `r` is bound to `15`.

(c) Finally, the sequence of expressions `g()` and `r` is evaluated. Since `g` was bound to `(fun () -> print_string "f\n")`, it causes the string `"f\n"` to be output, and since `r` was bound to `15`, the body of the function `f` evaluates to `15`.

5. Then the invocation of the function `f` mentioned in step 2 is completed. The formal parameters of `f` get bound to the arguments `(fun () -> print_string "e\n")` and `15` (since the operations in step 4 evaluated to `15`). In the body of `f`, the value of the `if-then-else` expression is evaluated and bound to `r` as follows:

   (a) the expression `(print_string "a"; x > 5) && (g(); x > 10)` in the `if` condition is first evaluated. The two arguments of the `&&` operator in this expression are `(print_string "a"; x > 5)` and `(g(); x > 10)`. Since `&&` use short-circuit evaluation, the left argument is evaluated first, which causes the string `"a"` to be output and evaluates to `true` since `15 > 5`. Then the right argument of `&&` is evaluated, which results in invoking the anonymous function `fun () -> print_string "e\n"` and thus the string `"e\n"` is output; and the sequence evaluates to `true` since `15 > 10`. Therefore, the `if` condition evaluates to `true`.

   (b) The expression under the `then` part is then evaluated. This results the string `"b"` being output and evaluates to the value `15 - 7 = 8`. Thus `r` is bound to `8`.

   (c) Finally, the sequence of expressions `g()` and `r` is evaluated. Since `g` was bound to `fun () -> print_string "e\n"`, it causes the string `"e\n"` to be output, and since `r` was bound to `8`, the body of the function `f` evaluates to `8`.

6. The variable `u` thus gets bound to the value `8`, which is output by the top-level OCaml loop.

2 Machine Problems (40 points)

Problems

Note: There are multiple ways to solve a problem. The solutions show one such method.

Problem 1: (2 point)

Write a function `rev_apply` which takes a function `f` and a pair `(x,y)`, interchanges `x` and `y` and applies the function `f` to both.

```
# let rev_apply f (x, y) = ...;;
val rev_apply : ('a -> 'b) -> 'a * 'a -> 'b * 'b = <fun>
# rev_apply (fun n -> n + 1) (2, 3);;
- : int * int = (4, 3)
```

Solution:

```
let rev_apply f (x,y) = (f y, f x);;
```

Problem 2: (4 points)

Consider the following mathematical definition of a sequence s_n:

$$s_n = \begin{cases} 1 & \text{if } n \leq 1 \\ 3 * s_{\frac{n}{2}} & \text{if } n \text{ is even} \\ 2 + s_{n-1} & \text{if } n \text{ is odd} \end{cases}$$

Write a recursive function `s` that takes an integer n and returns an integer s_n.

```
# let s n = ...;;
val s : int -> int = <fun>
# s 9;;
- : int = 29
```

Solution:

```
let rec s n =
if n <= 1 then 1
else if n mod 2 == 0 then (3 * s (n/2))
else (2 + s (n-1));;
```

Problem 3: (5 points)

Run-length encoding (RLE) is a data compression technique in which maximal (non-empty) consecutive occurrences of a value are replaced by pa pair of the value and a counter showing how many times the value was repeated in that consecutive sequence. For example, RLE would encode the list `[1;1;1;2;2;2;3;1;1;1]` as: `[(1,3);(2,3);(3,1);(1;3)]`. Write a function `rle` that takes a list and encodes it using the RLE technique. The function is required to use (only) forward recursion (no other form of recursion). You may not use any library functions.

```
# let rec rle lst = ... ;;
val rle : 'a list -> ('a * int) list = <fun>
# rle [1;1;1;2;2;2;3;1;1;1];;
- : (int * int) list = [(1, 3); (2, 3); (3, 1); (1, 3)]
# rle ['a';'b';'a';'a';'a';'c'];;
- : (char * int) list = [('a', 1); ('b', 1); ('a', 3); ('c', 1)]
```

Solution:

```
let rec rle lst = match lst with
[] -> []
| (x :: xs) -> (match rle xs with
[] -> [(x, 1)]
| ((y,n)::rest) -> if x = y then ((y, (n+1))::rest) else (x,1)::(y,n)::rest);;
```

Problem 4: (5 points)

Write a function `merge` that takes two lists and returns a list. Assuming that the two input lists are sorted, the result should be a sorted list containing elements from both the lists.

```
# let rec merge l1 l2 = ...;;
val merge : 'a list -> 'a list -> 'a list = <fun>
# merge [1;2;5;6] [3;4;6;9];;
- : int list = [1; 2; 3; 4; 5; 6; 6; 9]
```

Solution:

```
let rec merge l1 l2 = match l1 with
[] -> l2
| x1::xs1 -> (match l2 with
[] -> l1
| x2::xs2 -> (if x1 < x2 then (x1:: merge xs1 l2) else (x2 :: merge l1 xs2)));;
```

Problem 5: (4 points)

Write a function `separate` that takes a list of integers and outputs a pair of lists with the first list containing all the odd integers in the original list and the second list contains all the even integers of the original list. The order of the integers in both lists must be the same as the original list.

```
# let rec separate l = ...;;
val separate : int list -> int list * int list = <fun>
# separate [1; 3; 2; 4; 5];;
- : int list * int list = ([1; 3; 5], [2; 4])
```

Solution:

```
let rec separate l = match l with
[] -> ([],[])
| x::xs -> (match separate xs with
(odd, even) -> if (x mod 2 = 0) then (odd, x::even) else (x::odd, even));;
```

Problem 6: (6 points)

Write a function `maxsumseq` that takes a list of integers and outputs the maximum sum of the elements in any subsequence. [*Hint 1*: To obtain maximum sum subsequence, a global sum can be kept for the list and updated if current sum is greater than the global sum at any point. The current sum can be reset to 0 when it is less than 0.] [*Hint 2*: You can define local (recursive, perhaps) function(s)]

```
# let maxsumseq l = ...;;
val maxsumseq : int list -> int = <fun>
# maxsumseq [-1; 3; 2; -2; 5; -16];;
- : int = 8
```

Solution:

```
let maxsumseq l =
let rec aux l c m = match l with
[] -> m
| x::xs -> (let s = x + c in
if s > 0 then aux xs s (if s > m then s else m) else aux xs 0 m)
in (aux l 0 0);;
```

Problem 7: (7 points)

Recall that a directed graph is a pair $G = (V, A)$ where

- V is a set of vertices or nodes,
- A is a set of ordered pairs of vertices, called arcs, directed edges, or arrows.

When we implements the directd graph, we usually use an initial interval of integers to represent V, and an adjacency matrix or list to represent the set A.

In this problem we will use an adjacency list to represent the set A. This means that we will use the integer list list to represent the adjacency list of a graph, and it has one list per node in the integer list list. The i^{th} position of the list contains all the nodes connected to the i^{th} node by an out-going edge, where the integers in each list represent the position of the corresponding node in the adjacency list. The labeling of nodes start from zero, and elements in the list are counted from zero. Write a function `check_adj` that takes an adjacency list and a pair of integers to check if there exists an edge from the first element to the second one. Remember that this is a directed graph. In this problem, you may use any library functions you choose. Also, the input integers in the lists will always be nonnegative, and you can safely assume that we will not test on any number which is bigger than the graph size minus one. However, if the input pair of integers contains a negative number, you should return false. Note OCaml's type inference may infer a more general type for your solution than the type listed below, so do not panic.

```
# let check_adj adj_list (a,b) = ...
val check_adj : 'a list list -> int * 'a -> bool = <fun>
# check_adj [[1;2;3;4];[3;0;4;5];[1;4;3;5];[2;1];[1;2];[2;3;4]] (0,3);;
- : bool = true
```

Solution:

```
let rec outnode_list l n = match l with
[] -> []
| (x::xs) -> if n < 0 then []
else if n = 0 then x else outnode_list xs (n-1);;

let rec elt x l = match l with
[] -> false
| (y::ys) -> (x = y) || elt x ys

let check_adj adj_list (a,b) =
a >= 0 && b >= 0 && elt b (outnode_list adj_list a);;
```

Problem 8: (7 points)

Write a function `cumsum` that takes a list of integers as input and returns a list of integers such that the ith element of the output list is the sum of all the elements from the beginning of the original

list to its ith element. You must only use tail recursion for this problem. You may not use any library functions except `@`.

```
# let cumsum l = ... ;;
val cumsum : int list -> int list = <fun>
# cumsum [1;2;3];;
- : int list = [1; 3; 6]
```

Solution:

```
let cumsum l =
let rec aux lst sum curr = match lst with
[] -> sum
| (x::xs) -> (aux xs (sum@[(x+curr)]) (x+curr))
in aux l [] 0;;
```