Problem 1: Type inference. Show the work of the type inference algorithm and the resulting function type (or a type mismatch, if any) for the following OCaml functions. Feel free to check your result in the OCaml interpreter.

1. let f x y = if x < 2 then x :: [] else x :: [y];
2. let f x y = x y;
3. let rec f = function
   [] -> []
   | x :: xs -> (not (x < 2)) :: f xs;
4. let f = function
   (x, []) -> x
   | (x, y) -> x + y;;
5. let f = function
   [] :: x -> x
   | y :: z -> y :: [] :: z
   | _ -> [];;

Additionally, please answer the following questions about the function in question 5:

- what is the type of each of the three empty lists in the function?
- what is the purpose of the last case in the pattern-matching?


Problem 3. In the following Java program please point out all L-values (expressions that are used to denote a memory location) and R-values (expressions used to denote a value in memory).

```java
import java.awt.*;

public class LRValues {

    public static void main(String[] args) {
        int x = 0;
        x++;
        boolean y = (x == 0);
        if (y) {
            y = !y;
        }
        int[] A = {1, 2, 3};
        for (int i = 0; i < 2; ++i) {
```
A[i] = A[i+1];
}
Point thePoint = new Point();
thePoint.x = thePoint.x + 1;
}