Describe OCaml’s facilities for treating functions as data, giving examples of their use in programs. Illustrate your answer by discussing the function `fold_right`:

```
let rec fold_right f l e
  match l with
  | [] -> e
  | x::xs -> f x (fold_right f xs e)
```

You have been asked to implement a data structure to represent family relationships. For each person, it should record his or her name, mother, father, and children. As a first attempt, you have been given the following variant type declaration:

```
type person = Person of string * person * person * person list
```

Identify two problems with this declaration that make it unusable. Modify the declaration to correct these problems.

Consider the following, simpler data structure for associating a person with his or her children:

```
type famtree = B of string * famtree list
```

Write an OCaml function that takes two arguments: a predicate `P` over family trees (a function of type `famtree -> bool`) and a family tree `t`. The result should be the list of all subtrees of `t` (possibly including `t` itself) satisfying the predicate. For full credit, give due attention to efficiency.