```
type name = string
type phone = int
type level = int

type description = phone * level
type register = (name * description) list

//1.1
let nrew = [("Joe",10101010,4);
            ("Sal",11111111,2);
            ("Sam",12121212,7);
            ("Jane",13131313,1);]

//1.2
let getPhone nm reg =
    let (n,p,l) = List.find (fun (x,y,z) -> x=nm) reg
    p;;

getPhone "Joe" nrew

//1.3
let delete nm reg = List.filter (fun (x,y,z) -> x <> nm) reg

delete "Joe" nrew

//1.4
let rec filterout = function
    | [] -> []
    | (x,y,z)::xs -> (x,y)::(filterout xs);;

let getCandidates lvl reg =
    let lis = List.filter (fun (x,y,z) -> (abs (lvl-z)) < 3) reg
    filterout lis;;

getCandidates 2 nrew

//2.1
type exp =  | C of int
            | BinOp of exp * string * exp


//C 1, x+x, x+1

//2.2
let rec toString = function
    | C i -> i.ToString()
    | BinOp (e1,s,e2) -> (toString e1) + s + (toString e2);;

//2.3
let rec makeOpList = function
    | C i -> []
    | BinOp (e1,s,e2) -> (makeOpList e1) @ [s] @ (makeOpList e2);;
```

```
53
54 let extractOps e = Set.ofList (makeOpList e)
55
56 //2.4
57 type exp =  | C of int
58             | BinOp of exp * string * exp
59             | Id of string
60             | Def of string * exp * exp
61
62 let rec search df = function
63     | C i -> true
64     | BinOp (e1,o,e2) -> (search df e1) && (search df e2)
65     | Id s -> List.contains s df
66     | Def (d,e1,e2) -> (search (d::df) e1) && (search (d::df) e2);;
67
68 let expr1 = Def("x", C 5, BinOp (Id "x", "+", Id "x"))
69 let expr2 = Def("x", C 5, BinOp (Id "y", "+", Id "x"))
70
71 search [] expr1;;//true
72 search [] expr2;;//false
73 search ["y"] expr2;;//true
74 //3.1
75 //f: (int * tree<'a>) -> tree<'a>)
76 //produces a new tree identical to the input tree
77 //but only to depth n
78
79 //g: ('a -> bool) -> tree<'a> -> tree<'a>
80 //Creates a new tree which includes the nodes for which P(a) holds.
81 //If a node does not satisfy P(a), the nodes two trees will not
82 //get included.
83
84 //h: ('a -> 'b) -> tree<'a> -> tree<'b>
85 //applies the function k to the value "a" in each of the tree's nodes.
86
87
```