```fsharp
// Solution to the exam set in 02157 Functional Programming
// 2020 May + 2021 May
//

//Problem 3

type T<'a> = | A of 'a
             | B of 'a * T<'a>
             | C of 'a * T<'a> * T<'a>
             | D of 'a * T<'a> list

//1 type T<int list * string>
let t1 = A([1;2;3], "a")
let t2 = B(([1;1;1], "b"), t1)
let t3 = C(([0;1;0], "c"), t1, t2)
let t4 = A([1;1],"d")
let t5 = D(([4;2;6], "e"), [t1; t2; t3; t4])

//3 ('a -> 'b) -> T<'a> -> T<'b>
let rec mapT f t =
    match t with
    |A v          -> A (f v)
    |B(v, t1)     -> B(f v, mapT f t1)
    |C(v, t1, t2) -> C(f v, mapT f t1, mapT f t2)

//4
let rec f (xs, s) = (List.sum xs, s)
mapT f t1
mapT f t2
mapT f t3
mapT f t4

//5 toSet: T<'a> * ('a -> bool) -> Set<'a>
let rec leaves t =
    match t with
    |A v        -> [v]
    |B(v,t1)    -> v::(leaves t1)
    |C(v,t1,t2) -> v::(leaves t1)@(leaves t2)
let toSet(t,p) = Set.ofList (List.filter p (leaves t))
toSet(t3,(fun (xs,s) -> List.sum xs > 0))

//7
let rec mapT1 f t =
    match t with
    |A v          -> A (f v)
    |B(v, t1)     -> B(f v, mapT f t1)
    |C(v, t1, t2) -> C(f v, mapT f t1, mapT f t2)
    |D(v, ts)     -> D(f v, mapAux f ts)
and mapAux f = function
               | [] -> []
               | t::ts -> (mapT1 f t)::(mapAux f ts)
mapT1 f t5

```

```
54  //7 method2
55  let rec mapT2 f t =
56      match t with
57      |A v          -> A (f v)
58      |B(v, t1)     -> B(f v, mapT f t1)
59      |C(v, t1, t2) -> C(f v, mapT f t1, mapT f t2)
60      |D(v, ts)     -> D(f v, List.map (mapT f) ts)  //  List.map (mapT f) ts: ↵
           (int list * string->int * string) -> T<int list * string> list ->    ↵
         T<int * string> list
61  mapT2 f t5
62
63  // Problem 2
64
65  let rec f x = function
66                | [] -> []
67                | y::ys -> (x,y)::f x ys
68  f "a" [1;2;3]
69
70  //5  tail-recursive variant based on an accumulating parameter
71  let rec f1 x acc = function
72                   | [] -> List.rev acc
73                   | y::ys -> f1 x ((x,y)::acc) ys
74  f1 "a" [] [1;2;3]
75
76  //6 continuation-based tail-recursive variant
77  let rec f2 x ys c =
78      match ys with
79      | [] -> c []
80      | y::ys -> f2 x ys (fun v -> c ((x,y)::v))
81  f2 "a" [1;2;2;3] id
82
83  //7 high-order function
84  let f3 x ys = List.foldBack (fun y rs-> (x,y)::rs) ys []
85  f3 "a" [1;2;2;3]
86
87
88  // Problem 1
89
90  type Tab<'a, 'b> = ('a * 'b list) list
91  let t1: Tab<int, string> = [(1, [ "a"; "b"; "c" ]); (4, [ "b"; "e" ])]
92
93  //1 isKey: 'a -> Tab<'a, 'b> -> bool
94  let rec isKey x = function
95                   | []                        -> false
96                   | (key, _)::ts when x=key -> true
97                   | t::ts                     -> isKey x ts
98  isKey 4 t1
99
100 //2 insert(x,y,t): 'a * 'b * ('a*'b list) -> ('a*'b list) list
101 let rec insert(x,y,t) =
102     match t with
103     | [] -> [(x, [y])]
104     | (key, ys)::ts when x=key -> (key, y::ys)::ts
```

```
105     | t::ts -> t::insert(x,y,ts)
106 insert(5,"a",t1)
107
108 //3 deleteKey:
109 let rec deleteKey x t =
110     match t with
111     | [] -> []
112     | (key, _)::ts when x=key -> ts
113     | t::ts -> t::(deleteKey x ts)
114 deleteKey 4 t1
115
116 //4 deleteElement y t
117 let rec deleteElement y t =
118     match t with
119     | [] -> []
120     | (key,ys1)::ts -> (key, List.filter (fun y' -> y<>y') ys1)::
          (deleteElement y ts)
121 deleteElement "e" t1
122
123 //5 fromPairs: ('a*'b) list -> Tab<'a,'b>
124 let fromPairs xs = List.fold (fun t (x,y) -> insert(x,y,t)) [] xs
125 fromPairs [(2,"c");(1,"a");(2,"b")]
126
127 // 2021 May
128 // Problem 5
129 let h f a b =
130     let b0 = b
131     let rec aux f a b =
132         match (a,b) with
133         |(a,_) when a<0  -> Seq.empty
134         |(a,b) when b<0  -> aux f (a-1) b0
135         |(a,b)           -> Seq.append (aux f a (b-1)) (Seq.ofList [(a,b,f
              (a,b))])
136     aux f a b
137 Seq.item 4 (h (fun(a,b) -> a + b) 1 2)
138
139 let h1 f a b = seq { for i in [0..a] do
140                          for j in [0..b] do
141                              yield (i,j,f(i,j))}
142 Seq.item 5 (h1 (fun(a,b) -> a + b) 1 2)
143 //
144 seq [1;2;4;6] // Okay
```