```
1  (* Problem 2 *)
2
3  (* Q1 *)
4
5  let timesTable n = seq {for i in [1..10] do
6                          yield n*i}
7
8  (* Alternative *)
9
10 let timesTable1 n = Seq.init 10 (fun i -> (i+1) *i);;
11
12 (* Q2 *)
13
14 let tableOf n m f = seq {for i in [1..n] do
15                           for j in [1..m] do
16                                yield (i,j,f i j) }
17
18 (* Alternative *)
19
20 let tableOf1 n m f =
21    let g i = Seq.fold (fun sq j -> Seq.append sq (Seq.singleton (i,j,f i j)))
       Seq.empty (seq [1..m])
22    Seq.collect g (seq [1..n]);;
23
24 (* Q3 *)
25
26 let aSeq = Seq.initInfinite (fun i -> String.replicate (i+1) "a");;
27
28 (* Alternative *)
29
30 let aSeq1 = seq { for j in Seq.initInfinite (fun i -> i+1) do
31                     yield (String.replicate j "a") }
32
33
34 (* Q4 *)
35
36 let rec f i = function
37               | []  -> []
38               | x::xs -> (x+i)::f (i*i) xs;;
39
40
41 (* The type of f: int -> int list - int list *)
42 (* f i [x0; x1; x2; ... ; xn ] = [x0+i; x1+i^2; x2+i^4;...; xn+i^(2^n)] *)
43
44
45 (* Q5 *)
46
47 let rec fA a i = function
48                  | []    -> List.rev a
49                  | x::xs -> fA (x+i::a) (i*i) xs;;
50
51
```

```
52
53 let rec fC k i = function
54                 | []    -> k []
55                 | x::xs -> fC (fun res -> k(x+i::res)) (i*i) xs;;
56
57
58
```