

```

1 //week3 simple company club
2 //=====Edition1 with List=====
3 (*type name = string
4 // telephone number :no
5 type no = string
6 // year of birth : yb
7 type yb = int list
8 // themes of interests : the themes of interest
9 type ths = string list
10 type description = no * yb * ths
11 type register = (name * description) list
12 type arrangement = (name * description) list // There maybe something wrong
13 //=====END=====
14 //=====2=====
15 let reg: register =
16   [ ("Jianan Alvin", ("597100728", [ 2001; 11; 11 ], [ "Math"; "study";
17     "AI" ]))
18     ("Wenjie Fan", ("91870584", [ 2001; 11; 11 ], [ "Algorithm"; "Backend";
19       "DM" ]))
20     ("K1", ("9999999", [ 2001; 11; 11 ], [ "soccer"; "study"; "AI" ]))
21     ("K2", ("8888888", [ 2001; 11; 11 ], [ "Math"; "jazz"; "AI" ]))
22     ("K3", ("7777777", [ 2001; 11; 11 ], [ "Math"; "soccer"; "jazz" ]))
23     ("K4", ("6666666", [ 2001; 11; 11 ], [ "jazz"; "study"; "AI" ]))
24     ("K5", ("5555555", [ 1980; 11; 11 ], [ "Math"; "study"; "soccer" ]))
25     ("K6", ("4444444", [ 1981; 11; 11 ], [ "soccer"; "study"; "jazz" ])) ]
26
27 let p1 (no: no, yb: yb, ths: ths) =
28   //printf "%d" yb.[0]
29   if yb.[0] <= 1982
30     && ((List.contains "soccer" ths)
31       && (List.contains "jazz" ths)) then
32     true
33   else
34     false
35
36 let p2 (no: no, yb: yb, ths: ths) =
37   if ((List.contains "soccer" ths)
38     || (List.contains "jazz" ths)) then
39     true
40   else
41     false
42
43 let rec test reg n p1 =
44   match reg with
45   | (c1, c2) :: tail ->
46     printfn "%d is %b" n (p1 c2)
47     test tail (n + 1) p1
48   | [] -> printf "end"

```

```

48 *)(*test reg 0 p1
49
50 test reg 0 p2*)(*
51
52 let extractTargetGroup p r =
53     let rec help p r =
54         match r with
55         | (c1, (no, yb, ths)) :: tail when p (no, yb, ths) -> (c1, no) :: (help p tail)
56         | [] -> []
57         | _ :: tail -> help p tail
58
59     help p r
60
61 printfn "%A" (extractTargetGroup p2 reg)
62 *)
63
64 //=====Edition 2 with map and set=====
65
66 type name = string
67 // telephone number :no
68 type no = string
69 // year of birth : yb
70 type yb = int list
71 // themes of interests : the themes of interest
72 type ths = string list
73 type description = no * yb * ths
74 type register_list = (name * description) list
75 type register_map = Map<name, description>
76 //=====END=====
77 //=====2=====
78 let reg: register_list =
79     [ ("Jianan Alvin", ("597100728", [ 2001; 11; 11 ], [ "Math"; "study"; "AI" ]))
80     ("Wenjie Fan", ("91870584", [ 2001; 11; 11 ], [ "Algorithm"; "Backend"; "DM" ]))
81     ("K1", ("9999999", [ 2001; 11; 11 ], [ "soccer"; "study"; "AI" ]))
82     ("K2", ("8888888", [ 2001; 11; 11 ], [ "Math"; "jazz"; "AI" ]))
83     ("K3", ("7777777", [ 2001; 11; 11 ], [ "Math"; "soccer"; "jazz" ]))
84     ("K4", ("6666666", [ 2001; 11; 11 ], [ "jazz"; "study"; "AI" ]))
85     ("K5", ("5555555", [ 1980; 11; 11 ], [ "Math"; "study"; "soccer" ]))
86     ("K6", ("4444444", [ 1981; 11; 11 ], [ "soccer"; "study"; "jazz" ])) ]
87
88 let reg_map = Map.ofList reg
89
90 let p1 (no: no, yb: yb, ths: ths) =
91     //printf "%d" yb.[0]
92     if yb.[0] <= 1982

```

```

93     && ((List.contains "soccer" ths)
94         && (List.contains "jazz" ths)) then
95         true
96     else
97         false
98
99 let p2 (no: no, yb: yb, ths: ths) =
100     if ((List.contains "soccer" ths)
101         || (List.contains "jazz" ths)) then
102         true
103     else
104         false
105
106 let rec test_list reg n p1 =
107     match reg with
108     | (c1, c2) :: tail ->
109         printfn "%d is %b" n (p1 c2)
110         test_list tail (n + 1) p1
111     | [] -> printf "end"
112
113 let extract_target_map reg_map p =
114     Map.filter (fun _ description -> p description) reg_map
115
116 let _ = extract_target_map reg_map p1
117
118 let _ = extract_target_map reg_map p2
119
120
121 (*test_list reg 0 p1
122
123 test_list reg 0 p2*)
124 (*
125 let extractTargetGroup p r =
126     let rec help p r =
127         match r with
128         | (c1, (no, yb, ths)) :: tail when p (no, yb, ths) -> (c1, no) :: (help p tail)
129         | [] -> []
130         | _ :: tail -> help p tail
131
132     help p r
133
134 printfn "%A" (extractTargetGroup p2 reg)*)
135

```