```fsharp
//week3 simple company club
//========================Edtion1 with List===============================
(*type name = string
// telephone number :no
type no = string
// year of birth  : yb
type yb = int list
// themes of interests : the themes of interest
type ths = string list
type description = no * yb * ths
type register = (name * description) list
type arrangement = (name * description) list // There maybe something wrong
//======================END================================
//======================2=================================
let reg: register =
    [ ("Jianan Alvin", ("597100728", [ 2001; 11; 11 ], [ "Math"; "study"; "AI" ]))
      ("Wenjie Fan", ("91870584", [ 2001; 11; 11 ], [ "Algorithm"; "Backend"; "DM" ]))
      ("K1", ("9999999", [ 2001; 11; 11 ], [ "soccer"; "study"; "AI" ]))
      ("K2", ("8888888", [ 2001; 11; 11 ], [ "Math"; "jazz"; "AI" ]))
      ("K3", ("7777777", [ 2001; 11; 11 ], [ "Math"; "soccer"; "jazz" ]))
      ("K4", ("6666666", [ 2001; 11; 11 ], [ "jazz"; "study"; "AI" ]))
      ("K5", ("5555555", [ 1980; 11; 11 ], [ "Math"; "study"; "soccer" ]))
      ("K6", ("4444444", [ 1981; 11; 11 ], [ "soccer"; "study"; "jazz" ])) ]

let p1 (no: no, yb: yb, ths: ths) =
    //printf "%d" yb.[0]
    if yb.[0] <= 1982
       && ((List.contains "soccer" ths)
           && (List.contains "jazz" ths)) then
        true
    else
        false

let p2 (no: no, yb: yb, ths: ths) =
    if ((List.contains "soccer" ths)
        || (List.contains "jazz" ths)) then
        true
    else
        false

let rec test reg n p1 =
    match reg with
    | (c1, c2) :: tail ->
        printfn "%d is %b" n (p1 c2)
        test tail (n + 1) p1
    | [] -> printf "end"

*)(*test reg 0 p1

test reg 0 p2*)(*

let extractTargetGroup p r =
    let rec help p r =
        match r with
        | (c1, (no, yb, ths)) :: tail when p (no, yb, ths) -> (c1, no) :: (help p tail)
        | [] -> []
        | _ :: tail -> help p tail

    help p r

printfn "%A" (extractTargetGroup p2 reg)
*)

//==========================Edition 2 with map and set=================================================

type name = string
// telephone number :no
type no = string
// year of birth  : yb
type yb = int list
// themes of interests : the themes of interest
type ths = string list
type description = no * yb * ths
type register_list = (name * description) list
type register_map = Map<name, description>
//=======================END================================
//========================2==================================
let reg: register_list =
    [ ("Jianan Alvin", ("597100728", [ 2001; 11; 11 ], [ "Math"; "study"; "AI" ]))
      ("Wenjie Fan", ("91870584", [ 2001; 11; 11 ], [ "Algorithm"; "Backend"; "DM" ]))
      ("K1", ("9999999", [ 2001; 11; 11 ], [ "soccer"; "study"; "AI" ]))
      ("K2", ("8888888", [ 2001; 11; 11 ], [ "Math"; "jazz"; "AI" ]))
      ("K3", ("7777777", [ 2001; 11; 11 ], [ "Math"; "soccer"; "jazz" ]))
      ("K4", ("6666666", [ 2001; 11; 11 ], [ "jazz"; "study"; "AI" ]))
      ("K5", ("5555555", [ 1980; 11; 11 ], [ "Math"; "study"; "soccer" ]))
      ("K6", ("4444444", [ 1981; 11; 11 ], [ "soccer"; "study"; "jazz" ])) ]

let reg_map = Map.ofList reg

let p1 (no: no, yb: yb, ths: ths) =
    //printf "%d" yb.[0]
    if yb.[0] <= 1982
```

```
        && ((List.contains "soccer" ths)
            && (List.contains "jazz" ths)) then
        true
    else
        false

let p2 (no: no, yb: yb, ths: ths) =
    if ((List.contains "soccer" ths)
        || (List.contains "jazz" ths)) then
        true
    else
        false

let rec test_list reg n p1 =
    match reg with
    | (c1, c2) :: tail ->
        printfn "%d is %b" n (p1 c2)
        test_list tail (n + 1) p1
    | [] -> printf "end"

let extract_target_map reg_map p =
    Map.filter (fun _ description -> p description) reg_map

let _ = extract_target_map reg_map p1

let _ = extract_target_map reg_map p2


(*test_list reg 0 p1

test_list reg 0 p2*)
(*
let extractTargetGroup p r =
    let rec help p r =
        match r with
        | (c1, (no, yb, ths)) :: tail when p (no, yb, ths) -> (c1, no) :: (help 
           p tail)
        | [] -> []
        | _ :: tail -> help p tail

    help p r

printfn "%A" (extractTargetGroup p2 reg)*)
```