```fsharp
//week5
open System
type CourseNo = int
type Title = string
type ECTS = int
type CourseDesc = Title * ECTS
type CourseBase = Map<CourseNo, CourseDesc>
type Mandatory = Set<CourseNo>
type Optional = Set<CourseNo>
type CourseGroup = Mandatory * Optional
type BasicNaturalScience = CourseGroup
type TechnologicalCore = CourseGroup
type ProjectProfessionalSkill = CourseGroup
type Elective = CourseNo -> bool
type FlagModel =
  BasicNaturalScience*TechnologicalCore*ProjectProfessionalSkill*Elective
type CoursePlan = Set<CourseNo>
//2015 summer
//1.
let isValidCourseDesc desc =
    match desc with
    | (_, ects) when ects % 5 = 0 -> true
    | (_, ects) -> false

(*isValidCourseDesc ("Computer Science Modelling", 5)*)
//2.
let isValidCourseBase cb =
    Map.forall (fun _ desc -> isValidCourseDesc desc) cb

(*isValidCourseBase (Map.ofList [ 2141, ("Computer Science Modelling", 4) ])*)
//3.
let disjoint s1 s2 =
    if (Set.intersect s1 s2) = Set.empty then
        true
    else
        false

(*disjoint (set [ 2131; 2141 ]) (set [ 2157; 2158 ])*)
//4.
let sumECTS cs cb =
    let p no cb = Map.containsKey no cb
    // let (no, ects) = Map.find no cb
    Set.fold
        (fun t no ->
            let (title, ects) = Map.find no cb
            if p no cb then ects + t else t)
        0
        cs

```

```
49 (*sumECTS
50     (set [ 2131; 2141 ])
51     (Map.ofList [ (2131, ("Embedded System", 5))
52                   (2141, ("Computer Science Modelling", 10)) ])*)
53 //5.
54 let isValidCourseGroup cg cb =
55     let (man,opt) = cg
56     let sumopt = sumECTS opt cb
57     let summan = sumECTS man cb
58     disjoint man opt
59     && (summan < 45 || (summan = 45 && Set.count opt = 0))
60     && summan + sumopt >= 45
61
62 (*isValidCourseGroup
63     (set [ 2131 ], set [ ])
64     (Map.ofList [ (2131, ("Embedded System", 45))
65                   (2141, ("Computer Science Modelling", 45))
66                   (2157, ("Founctional programming", 5))
67                   (2158, ("Parallel programming", 5))])*)
68 //6.
69 let union (man,opt) = Set.union man opt
70 let ep no = no > 2000
71 let isValid fm cb =
72     let (bns,tc,pps,ep) = fm
73     let allbns = union bns
74     let alltc = union tc
75     let allpps = union pps
76     let allcourses = Set.union (Set.union allbns alltc) allpps
77     printfn "run"
78     isValidCourseGroup bns cb
79     && isValidCourseGroup tc cb
80     && isValidCourseGroup pps cb
81     && Set.count allbns + Set.count alltc + Set.count allpps = Set.count
       allcourses
82     && Set.forall ep allcourses
83 (*isValid
84     ( (set [ 2131 ], set [ ]),(set [ 2141 ], set [2157 ]),(set [ 2158 ], set
       [1006 ]),ep )
85     (Map.ofList [ (2131, ("Embedded System", 45))
86                   (2141, ("Computer Science Modelling", 40))
87                   (2157, ("Founctional programming", 5))
88                   (2158, ("Parallel programming", 35))
89                   (2801, ("Introduction to Artificial Intelligence",5))
90                   (1006, ("Advanced Engineering mathematics1",10))])*)
91 //7.
92 let checkPlan cp fm cb =
93     let (bns,tc,pps,ep) = fm
94     sumECTS cp cb = 180
95     && sumECTS (union bns) cb = 45
```

```
96      && sumECTS (union tc) cb = 45
97      && sumECTS (union pps) cb = 45
98  checkPlan
99      ( set [2131;2141;2157;2158;1006;2800] )
100     ( (set [ 2131 ], set [ ]),(set [ 2141 ], set [2157 ]),(set [ 2158 ], set
          [1006 ]),ep )
101     (Map.ofList [ (2131, ("Embedded System", 45))
102                   (2141, ("Computer Science Modelling", 40))
103                   (2157, ("Founctional programming", 5))
104                   (2158, ("Parallel programming", 35))
105                   (2801, ("Introduction to Artificial Intelligence",45))
106                   (1006, ("Advanced Engineering mathematics1",10))])
107
108
```