1 Opgave

1.1

There are a number of ways to make a 2 dementional vector i f shape one of the ways is to make a tuple.

1.2

I am giving the funtions the following names: 1. Lenghtofvector 2. Addofvector 3. Skaleofvector Since names shoud give an indication of what the fucktions do.

1.3

In the function "Lenghtofvector", the function takes as input a tuple with the type of floats and outputs a float. The function takes each element of the tuple and then puts it in the mathematical form for the length of a vector.

In the function "Addofvector" takes as input a 2 tuple with the type of floats. It outputs one tuple that is each of the first elements of the input added together and each of the second element from the input added together.

In the function "Skaleofvector" takes as input a tuple and a float. It outputs a tuple where each element has been multiplied with the float giving in the input.

1.4

if i define 2 vectors as follows

```
let x = (2.0,3.0)

let y = (1.0,2.0)
```

the output will be, 3,6055 for the first function. (3.0,5.0) for the second and (4.0,6.0) for the last function.

1.5

```
let x = (2.0,3.0)

let y = (1.0,2.0)

//printfn "%A" (fst x)


let lenghtofvector vector:float = List.length(vector)
```

```
printfn "%d" (lenghtofvector x)


let addofvector (vector1:float * float) (vector2:float * float) =
    (fst vector1+ fst vector2, snd vector1+ snd vector2)
printfn "%A" (addofvector x y )


let skaleofvector (vector: float * float)  (skale:float) =
   (fst vector * skale, snd vector * skale)
printfn "%A" (skaleofvector x 2.0)
```

Using kens method for function design. The function should return the lenght of the vector. I have given the name lenghtofvector, since it is a good describtion of what the function does. it takes as a input a list, and returns a float.

The function should return the two vectors added together. I have given the name addofvector, since it is a good describtion of what the function does. it takes as two lists as input, and returns a list where each element have been added together.

The last funtion should return a vector multiplied with a skala. I have given the name skaleofvector, since it is a good describtion of what the function does. The input is a list and a skala, the output is a list where each element have been multiplied with the skala.

1.6

if i wanted it in 3 demention, i would just add another element to the tuple. like this $x = (2.0, 3.0, 4.0)$

2

2.1

One way is to make a list of tuplis, each tuplis have a studentid, and a list of ansawers.

```
let studentsSurveys = [
    (1, ["A"; "B"; "C"; "D"])
    (2, ["B"; "A"; "D"; "C"])
    (3, ["C"; "C"; "B"; "A"])
    (4, ["D"; "B"; "A"; "D"])
]
```

another way is just using a dictionary

```
let studentsSurveys: Survey list = [
    { StudentId = 1; Answers = ["A"; "B"; "C"; "D"] }
    { StudentId = 2; Answers = ["B"; "B"; "B"; "C"] }
    { StudentId = 3; Answers = ["C"; "B"; "B"; "A"] }
    { StudentId = 4; Answers = ["D"; "B"; "B"; "D"] }
```

2.2

Using kens method for function design. The function should count the number of students that have given a specifc answer to a given question. I have given the name countAnswer, since it is a good describtion of what the function does.

```
let countAnswer(studentsquiz: (int * string list) list) questionIndex answer
=
    studentsquiz
    |> List.filter (fun (_, answers) -> answers.[questionIndex] = answer)
    |> List.length
```

```
let counta =  countAnswer studentsquiz 0 "A"
let countb =  countAnswer studentsquiz 0 "B"
let countc =  countAnswer studentsquiz 0 "C"
let countd =  countAnswer studentsquiz 0 "D"
```

Since the function is supposed to count the answer, I have given it the name of countanswer. As an input it takes the student quiz and a qustionindex from 0-3, and a specific ansawer from A..D. First part takes as input filters only students that have ansawed the specific ansawer to the specific question. After that the list.length counts the length of this new list. witch in our case is the number of students.

2.3

Since the function is supposed to count the question in procent, I have given it the name of countquestionsprocent. As an input it takes the an index from 0-3. First it definds 4 let bindings where it uses the function from before, and then changes the answer from A to D. Then 4 new let binding that define the procent for each anwaser. and divided with the total amount in this case 4. And then it prints the results, and you have to do this for each question.

```
let countquestionsprocent indexqueston =
```

```
    let counta =  countAnswer studentsquiz indexqueston "A"
    let countb =  countAnswer studentsquiz indexqueston "B"
    let countc =  countAnswer studentsquiz indexqueston "C"
    let countd =  countAnswer studentsquiz indexqueston "D"
    let (procenta:float) = float counta/4.0
    let (procentb:float) = float countb/4.0
    let (procentc:float) = float countc/4.0
    let (procentd:float) = float countd/4.0
    printfn "Answer A: %A  Answer B: %A Answer c:  %A Answer D: %A"
procenta procentb procentc procentd

//printfn "procents to question 1%A" (countquestionsprocent 0 )
//printfn "procents to question 2%A" (countquestionsprocent 1 )
//printfn "procents to question 3%A" (countquestionsprocent 2 )
//printfn "procents to question 4%A" (countquestionsprocent 3 )
```

2.4

Since the function is supposed to check if any of the students have ansawer the same, I have given it the name of compare since it explains what it does. First i make 4 let bindings each, takes the ansawers of the 4 students. After that it compares each list of ansawers with each orther and retrun true if they are equal with each orther. After that there is 7 if else statements that only write if the condition is true.

```
let firstSurveyAnswers = studentsquiz |> List.item 0 |> snd
let secondSurveyAnswers = studentsquiz |> List.item 1 |> snd
let thirdSurveyAnswers = studentsquiz |> List.item 2 |> snd
let fouthSurveyAnswers = studentsquiz |> List.item 3 |> snd

let compare =
    let s1s2 = firstSurveyAnswers = secondSurveyAnswers
    let s1s3 = firstSurveyAnswers = thirdSurveyAnswers
    let s1s4 = firstSurveyAnswers = fouthSurveyAnswers
    let s2s3 = secondSurveyAnswers = thirdSurveyAnswers
    let s2s4 = secondSurveyAnswers = fouthSurveyAnswers
    let s3s2 = thirdSurveyAnswers = secondSurveyAnswers
    let s3s4 = thirdSurveyAnswers = fouthSurveyAnswers

    if s1s2 = true then printfn "s1s2 have same ansawer" else printfn ""

    if s1s3 = true then printfn "s1s3 have same ansawer" else printfn ""

    if s1s4 = true then printfn "s1s4 have same ansawer" else printfn ""
```

```
    if s2s3 = true then printfn "s2s3 have same ansawer" else printfn ""
    if s2s4 = true then printfn "s2s4 have same ansawer" else printfn ""
    if s3s2 = true then printfn "s3s2 have same ansawer" else printfn ""
    if s3s4 = true then printfn "s3s4 have same ansawer" else printfn ""
printfn"comparefunction %A" compare
```

3

3.1

A function have a name, can take parameters, and have a body that contains the code to be executed.

3.2

Type inference is a feature that allows the compiler to automatically deduce the types of expressions without explicit types. This makes the code more concise and readable.

3.3

Functional programming offers some advantages it makes the code more robust, maintainable, and efficient. In my experince it makes it easy to understand what the progrom does if the correct functionsnames is define.