Advanced Programming (I00032)
An n-person idea box in iTasks

Assignment 5

In this assignment you create a number of versions of an idea box using the task combinators explained in the lecture notes. You have to hand in only the final version of this program. Although the final version of the task contains a lot of possibilities to code can be quite concise: about 60 lines of code.

1 Basis

Create an iTask that enables the user to edit a list of ideas. An idea can for instance be represented as a record of type `Idea` like:

```
:: Idea =
    { idea      :: String
    , details   :: Maybe Note
    , user      :: Name
    , number    :: Int
    }
:: Name :== String
```

2 Persistent Memory for Ideas

It is more convenient to remember ideas between sessions and to share them with other users. This is achieved by a shared store like:

```
ideas :: Shared [Idea]
ideas = sharedStore "Ideas" []
```

Change you program such that the list of ideas is stored in the store `ideas`. Using `updateSharedInformation` the users can update this store.

The iTask system stores shared information in a subfolder in the same directory as the executable. Delete this subfolder to empty or remove the store. When you change the type of the store it is highly recommended to remove this subfolder.

3 Editing a Single Idea

Instead of editing the whole list of ideas the user should now be able to edit only a single idea. When she is done the idea is added to the list of ideas in the store.

Using a `forever` the task `addNewIdea` is repeated infinitely often.

4 Add Name and Number Automatically

It is rather cumbersome for the user to enter her name and the number of each idea. Change you program such that your program asks the name of the user just once and

adds the name to the `idea` and `details` entered by the user. The number of the idea can be found by counting the number of ideas in the store.

5 Multiple Users

Since the ideas are stored in a shared store, they can be used by multiple users. Observe what happens when you run your program simultaneously in different browser windows or in different tabs of the same window.

6 Viewing Existing Ideas

In the current version of your program the user can see only the idea that she is editing. It is of course convenient to see also the other ideas. Using a parallel composition, e.g. with `-||-`, of a view task and your add new idea task the user can see the existing ideas and add a new idea at the same time. Use `enterChoiceWithShared` to show a table with all existing ideas.

7 Displaying the Selected Idea

When the user selects an idea it can be shown in more detail by adding a subexpression like `>&^ viewSharedInformation` to the choice of the previous step.

8 Actions

Add the following actions to displaying of elements and a current element:

1. An `ok` button to undo the selection of the current idea.
2. A button labelled `Delete All` that clears the list of ideas. It is always enabled.
3. A button to delete the current idea. This is of course only enabled if there is an idea selected. Moreover, users can only delete their own ideas with this button.

   Depending on how you identify ideas and assign numbers to them, it might be necessary to renumber the remaining ideas after deleting an idea.
4. A `Like` button to express sympathy for an idea. This button can only be used when there is an idea selected. Moreover, user cannot use this button for their own ideas.

   The easiest solution to store the likes of an idea seems to extend the record `Idea` with an appropriate field.

Deadline

The deadline for this exercise is October 12, 13:30.