Advanced Programming (I00032)
An n-person chat application in iTasks

Assignment 5

In this assignment you create two new versions of task combinators / applications that have been explained in the lecture notes.

1 `allTasks`, revisited

In the lecture slides the derived iTask combinator `allTasks` has been explained (slide 27). The version as presented in the lecture (which is exactly the same as the one found in module *CommonCombinators.icl*) is a task with a task value that holds all *current* sub-task values, regardless whether they are stable or not. Tasks without task value are of course neglected in this result task value list. As a consequence, this combinator has the weak property that the length of the list of its task value is less or equal to the length of task arguments with which `allTasks` is called.

Define a new version of `allTasks`, called `reallyAllTasks`, that has the same signature as `allTasks` but that has the stronger property that the length of the list of task values is *exactly the same* as the length of the list of task arguments with which `reallyAllTasks` is called.

2 An n-person chat application

In the lecture slides a 2-person chat application has been presented (slides 31 and 32). In this assignment you create a new iTask application that allows an *arbitrary* number of users to chat with each other. Do this in two stages, described below.

2.1 n is fixed

The chat application presented in the lecture started with the selection of 2 users that are going to chat. Replace this functionality with the option to choose an arbitrary number of users that remains fixed during the entire chat session.

2.2 n is dynamic

Adapt your generalized chat application in such a way that during a chat session users can enter and leave the discussion. Use for this purpose the task functions `appendTask` and `removeTask` that have been introduced on slide 25.

3 Deadline

The deadline for this exercise is October 14, 23:59.