advanced programming
tutorial 6

October 19, 2018

Pieter Koopman

Radboud University

questionnaires

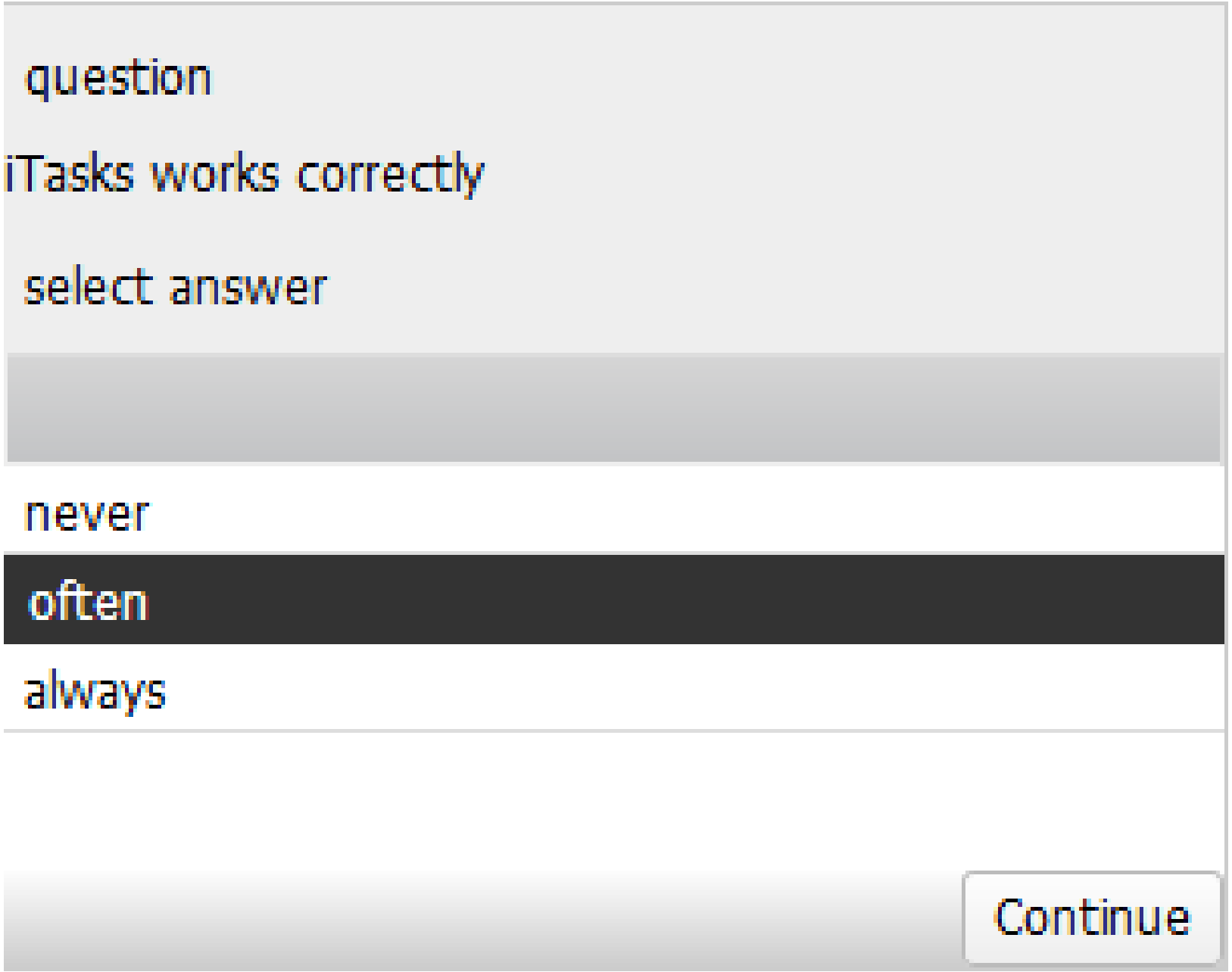
```
:: Question =
   {question :: String
   ,answers  :: [String]
   ,correct  :: Int
   }

derive class iTask Question

questions :: Shared [Question]
questions = sharedStore "questions" []
```

main task

```
mainTask :: (String, Function) -> Task [Question]
mainTask (user, role) =
   case role of
      Student = studentTask user
      Teacher = teacherTask user
      Admin   = updateSharedInformation "Admin view" []
questions
      _       = abort "unknown role"
```

name task

```
nameTask :: ((String, Function) -> Task a) -> Task a |
iTask a
nameTask task
 =    (enterInformation "enter your name" [] -&&-
        updateInformation "select role" [] Student)
  >>* [OnAction ActionOk
                (ifValue (\(name, role).size name > 2)
task)
      ]
```



student task

student task

```
studentTask name
 = get questions >>= \list. answer list 0 0  where
 answer [] correct wrong
  = viewInformation (name + " answered " + toString correct +
    " questions correct and " + toString wrong + " questions wrong")
    [] []
 answer [q:rest] correct wrong
  = viewInformation "question" [] q.question
    ||- editChoice "select answer"
        [ChooseFromGrid snd] [(i,a) \\ i <- [1..] & a <- q.answers]
Nothing
   >>= \(i, a).
   if (i == q.correct)
   (answer rest (correct + 1) wrong)
   (answer rest correct (wrong + 1))
```

teacher task

Choose an item to edit

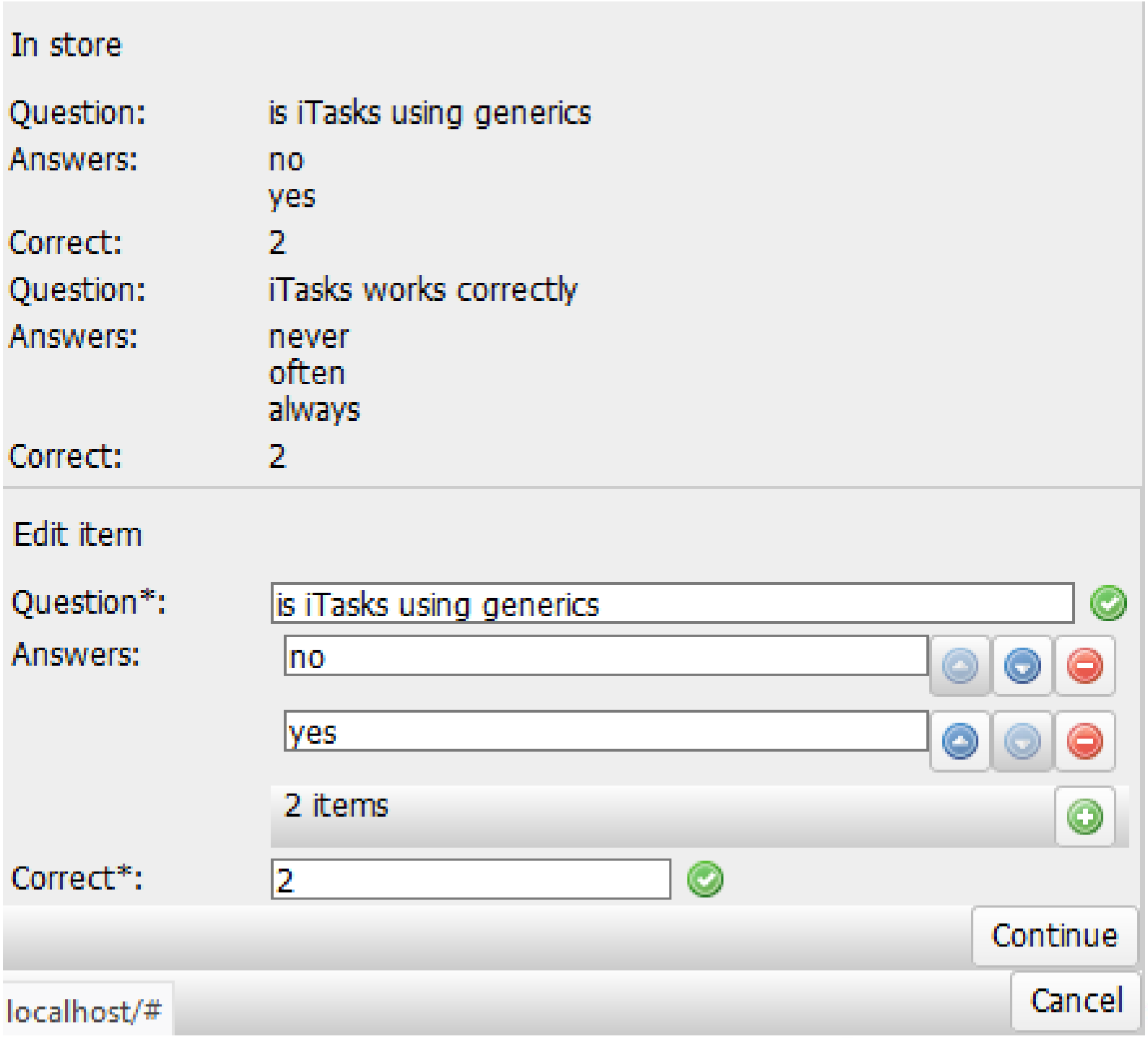
Question	Answers	Correct
is iTasks using generics	[no, yes]	2
iTasks works correctly	[never, often, always]	2

Append | Delete | Edit | First | Clear | Quit

teacher task

```
editSharedList :: (Shared [a]) -> Task [a] | iTask a
editSharedList store
 = enterChoiceWithShared "Choose an item to edit"
     [ChooseFromGrid snd]
     (mapRead (\ps -> [(i,p) \\ p <- ps & i <- [0..]]) store)
  >>*  [ OnAction (Action "Append") (hasValue (showAndDo append))
     , OnAction (Action "Delete") (hasValue (showAndDo delete))
     , OnAction (Action "Edit")   (hasValue (showAndDo edit))
     , OnAction (Action "First")  (always (showAndDo first undef))
     , OnAction (Action "Clear")  (always (showAndDo append
                                                  (-1,undef)))
     , OnAction (Action "Quit")   (always (get store))
     ]
```

show and do


In store
Question:
is iTasks using generics
Answers:
no
yes
Correct:
2
Question:
iTasks works correctly
Answers:
never
often
always
Correct:
2
Edit item
Question*:
is iTasks using generics
Answers:
no
yes
2 items
Correct*:
2
Continue
Cancel
localhost/#

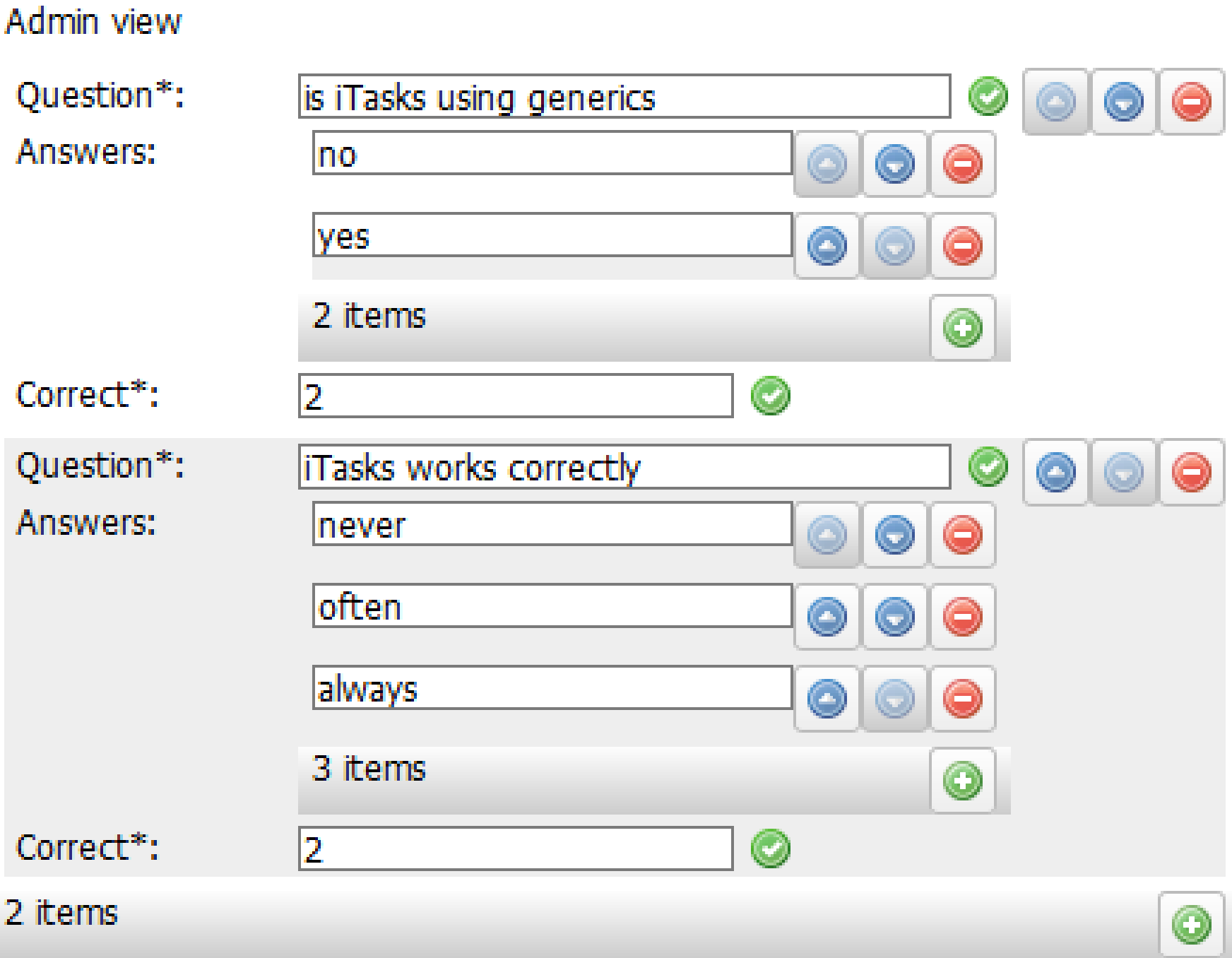
teacher task 2

```
showAndDo fun ip
 = viewSharedInformation "In store" [] store
   ||-  fun ip
   >>* [ OnValue (hasValue  (\_ -> editSharedList store))
       , OnAction (Action "Cancel") (always  (editSharedList
store))       ]
append (i,_)
 = enterInformation "Add new item" []
  >>=  \n . upd (\ps .
       let (begin, end) = splitAt (i + 1) ps
        in begin ++ [n] ++ end) store
```

teacher task 3

```
delete (i, _)
 = upd (removeAt i) store
first _
 = enterInformation "Add new item" []
   >>= \n . upd (\l . [n: l]) store
edit (i, p)
 = updateInformation "Edit item" [] p
   >>= \p . upd (\ps .  updateAt i p ps) store
```

admin


Admin view
Question*:
is iTasks using generics
Answers:
no
yes
2 items
Correct*:
2
Question*:
iTasks works correctly
Answers:
never
often
always
3 items
Correct*:
2
2 items

GETTING THE WORK DONE

improving your final mark

- this assignment can improve your final mark for this course
  - no effect if the exam result is better than this assignment
- only if exam result ≥ 5
- weight of this assignment will be 25%
- it must be your own work
  - working with a partner is allowed

- just a reward to stimulate you to make the assignment

- **no lectures in the next two weeks**