The University of Melbourne
School of Computing and Information Systems
COMP90048
Declarative Programming
First Semester, 2019

assignment1
Project Specification
Due Tuesday, 26 March 2019, 4:00PM
Worth 5%

The objective of this project is to practice your Haskell programming skills. You will write a few fairly simple Haskell functions.

The Assignment

You will implement the following Haskell functions.

1. `subst :: Eq t => t -> t -> [t] -> [t]`
   `subst` takes two values and a list, and replaces every occurrence of the first value with the second in the list. For example:

`subst 0 1 [0,1,2,3]`	should return	`[1,1,2,3]`
`subst 'e' 'o' "dog"`	should return	`"dog"`
`subst 'e' 'o' "elephant"`	should return	`"olophant"`

2. `interleave :: [t] -> [t] -> [t]`
   `interleave` takes two lists and returns the interleaving of the two arguments. That, the result is a list in which the first, third, fifth ... elements come fromt the first argument and the second, fourth, sixth ... come from second. If either argument is shorter than the other, the excess elements of the longer comprise the end of the resulting list. For example:

`interleave [1,2,3,4] [11,12,13,14]`	should return	`[1,11,2,12,3,13,4,14]`
`interleave "" "dog"`	should return	`"dog"`
`interleave "wl" "arus"`	should return	`"walrus"`
`interleave "tlpone" "eeh"`	should return	`"telephone"`

3. `unroll :: Int -> [a] -> [a]`
   `unroll` takes a list and an integer and constructs a list of the specified length made up by "unrolling" the input list as many times as needed to construct a list of that length. That is, the output consists of the input list repeated as many times as necessary to have the specified length. For example:

`unroll 3 [1,2,3,4,5]`	should return	`[1,2,3]`
`unroll 8 [1,2,3]`	should return	`[1,2,3,1,2,3,1,2]`
`unroll 4 "ski"`	should return	`"skis"`

You must call your source file `Assignment1.hs` or `Assignment1.lhs` (note the capitalisation), and it must begin with the module declaration:

```
module Assignment1 (subst, interleave, unroll) where
```

I will post on the LMS a test driver program called `Assignment1Test` which is substantially similar to the test driver I will use for testing your code. I will compile and link your code for testing using the following command (or similar):

```
ghc -O2 --make Assignment1Test
```

Assessment

Your project will be assessed 100% on correctness. For this assignment, code quality will not be considered. However, for your own sanity, I do recommend commenting your code and programming it carefully, paying due attention to programming technique.

Note that timeouts will be imposed on all tests. Test cases will be rather small, so the timeouts should only affect you if you create an infinite recursion (infinite loop).

Submission

You **must** submit your project from either of the unix servers `dimefox.eng.unimelb.edu.au` or `nutmeg.eng.unimelb.edu.au`. Make sure the version of your program source files you wish to submit is on this host, then `cd` to the directory holding your source code and issue the command:

```
submit COMP90048 assignment1 Assignment1.hs
```

(or substitute `Assignment1.lhs` for `Assignment1.hs` if you are writing in literate Haskell).

Important: you must wait a minute or two (or more if the servers are busy) after submitting, and then issue the command

```
verify COMP90048 assignment1 | less
```

This will show you the test results from your submission, as well as the file(s) you submitted. If the test results show any problems, correct them and submit again. You may submit as often as you like; only your final submission will be assessed.

If you wish to (re-)submit after the project deadline, you may do so by adding "`.late`" to the end of the project name (*i.e.,* `assignment1.late`) in the `submit` and `verify` commands. But note that a penalty, described below, will apply to late submissions, so you should weigh the points you will lose for a late submission against the points you expect to gain by revising your program and submitting again.

It is your responsibility to verify your submission.

Your submission will be tested on one of the servers `dimefox2.eng.unimelb.edu.au` or `nutmeg2.eng.unimelb.edu.au` (note the "2"). These servers run GHC version 8.4.3. You are advised to test your program on one of these servers before submitting; in the unlikely case that your program uses some Haskell features not supported by GHC 8.4.3, it will be much easier to discover this.

Note that these hosts, as well as the hosts you must use for submission, are only available through the university's network. If you wish to use these machines from off campus, you will need to use the university's Virtual Private Network. The LMS Resources list gives instructions.

Windows users should see the LMS Resources list for instructions for downloading the (free) MobaXterm or Putty and Winscp programs to allow you to use and copy files to the department servers from windows computers. Mac OS X and Linux users can use the `ssh`, `scp`, and `sftp` programs that come with your operating system.

Late Penalties

Late submissions will incur a penalty of 0.5% of the possible value of that submission per hour late, including evening and weekend hours. Late submissions will incur a penalty of 0.5% per hour late, including evening and weekend hours. This means that a perfect project that is much more than 4 days late will receive less than half the marks for the project. If you have a medical or similar compelling reason for being late, you should contact the lecturer as early as possible to ask for an extension (preferably before the due date).

Note Well:

This project is part of your final assessment, so cheating is not acceptable. Any form of material exchange between teams, whether written, electronic or any other medium, is considered cheating, and so is the soliciting of help from electronic newsgroups. Providing undue assistance is considered as serious as receiving it, and in the case of similarities that indicate exchange of more than basic ideas, formal disciplinary action will be taken for all involved parties. If you have questions regarding these rules, please ask the lecturer.