Assignment 3 Report

Name:	*Wei XING (Phillip)*
Student Number:	*U5656487*
Course:	*COMP1100*
Date:	*25/05/2017*
Collaborators: help partner:	*Yian Hu (u5842618), Shukai Zhang (u5686922)*
tutor:	*Yiping Su (u5925716)*
event:	*PAL, COMP1100 Study Event on Wednesday*

Introduction

Sudoku is a puzzle played on a partially filled 9×9 grid. The assignment 3 is to complete all the "*TODO*" functions and finally using numbers from 1 to 9 such that the entries in each row, each column and each major 3×3 block are pairwise different. This assignment teaches us how to find a unique solution without guessing, i.e. without search. I have finished "solve" function using *backtracking* and my own solution.

1. allBalnks *(pass the test)*

The hints in the assignment introduction prompt to use **replicate** function, after understanding this function and according to the Sudoku type, I need to create a list of lists, which means I need to use **replicate** function twice.

2. isSudoku *(pass the test)*

The hint in the assignment introduction prompt to use **all** and **length** function, similar like above, after understanding these two functions and according to Bool type, I need to check whether Sudoku contains nine lists in its list, which I use case statement to make a judgement.

3. noBlanks *(pass the test)*

This function asks me to check if Sudoku list contains "Nothing" element, I use "notElem" function to check whether one element is in the list, since Sudoku list is a list of lists, which I need to extract outside list and put all elements into one list to fit the request of notElem function, so I use concat function.

4. printSudoku *(pass the test)*

I get some ideas and hints from Stackoverflow[1] and know the meaning of putStr and unlines function. As taught in the lecture before, I need to use ASCII table to make a transfer between char and Int. Since putStr needs String and unlines extracts the

[1] *https://stackoverflow.com/questions/41512822/error-when-changing-function-name-to-list-haskell*

String out of the list, so I write a map function to transfer Sudoku to String, which will also be used in the following function.

5. fromString

Firstly, since String is actually [Char], I write a map function to list the Cell so that it can fit the type of Sudoku, but list the Cell just one list, I still need to add another list outside, so I make another helper function to list 9 elements in a list and recursion it to form a whole list[2].

6. toString

I use concat function to put all the elements in Sudoku into one list to make it become Cell type, then I use the convertInt function before to map all the elements in the list to make it form a String.

```
-- >>> toString example
-- "36..712...5....18...92.47......13.284..5.2..927.4
6......53.89...83....6...769..43"
-- >>> fromString (toString example)
-- Sudoku [[Just 3,Just 6,Nothing,Nothing,Just 7,Just 1,Just
 2,Nothing,Nothing],[Nothing,Just 5,Nothing,Nothing,Nothin
g,Nothing,Just 1,Just 8,Nothing],[Nothing,Nothing,Just 9,Jus
t 2,Nothing,Just 4,Just 7,Nothing,Nothing],-- [Nothing,Nothi
ng,Nothing,Nothing,Just 1,Just 3,Nothing,Just 2,Just 8],[Jus
t 4,Nothing,Nothing,Just 5,Nothing,Just 2,Nothing,Nothing,Ju
st 9],[Just 2,Just 7,Nothing,Just 4,Just 6,Nothing,Nothing,N
othing,Nothing],-- [Nothing,Nothing,Just 5,Just 3,Nothing,Ju
st 8,Just 9,Nothing,Nothing],[Nothing,Just 8,Just 3,Nothing,
Nothing,Nothing,Nothing,Just 6,Nothing],[Nothing,Nothing,Ju
st 7,Just 6,Just 9,Nothing,Nothing,Just 4,Just 3],[]]
```

7. rows and cols

Following the assignment instruction and hints, understanding the function transpose in the hints, I think these two functions are not very hard.

```
-- >>> rows [[1,2,3,4,5],[5,6,7,8],[9,10,11,12,13]]
--[[1,2,3,4,5],[5,6,7,8],[9,10,11,12,13]]
-- >>> cols [[1,2,3,4,5],[5,6,7,8],[9,10,11,12,13]]
-- [[1,5,9],[2,6,10],[3,7,11],[4,8,12],[5,13]]
```

8. boxs

Firstly, I write a `group3` function which groups a list by three elements, from the StackOverFlow[3] hints, it asks me to trace out the execution - first look at `map group3 boxes`, then `transpose $ map group3 boxes`, then `concat $ transpose $ map group3 boxes`.

[2] https://stackoverflow.com/questions/41126330/writing-sudoku-in-haskell-find-possible-candidates-for-a-cell

[3] https://stackoverflow.com/questions/31360775/find-9-3x3-blocks-of-sudoku-board-haskell

I follow this guidance and get my boxs at last.

```
-- >>> boxs [[1,2,3,4,5],[5,6,7,8],[9,10,11,12,13],[14,15,16,17,18]]
-- [[1,2,3,5,6,7,9,10,11],[14,15,16,4,5,8],[12,13,17,18]]
```

9. okBlock and okSudoku *(pass the test)*

Block Cell is *[Maybe Int]* and again use *notElem* function to compare and use recursion keep judging the rest of the list. The tutor *Yiping Su* reminds me to use all function in okSudoku and compare the meaning of written okBlock function and the meaning of Sudoku (I have written in the introduction), I get this function.

10. blank *(pass the test)*

After the instruction of assignment[4], I notice that if position number divide 9, the quotient number is the front position in the tuple and remainder number is the back position in the tuple. So, I need to extract *Int* in *Maybe Int*, and write a helper function.

11. (!!=) *(pass the test)*

This function just extract the specific position in the input list and keep the rest, I know how to use *take* and *drop* function, so this is not very hard for me.

12. update (*test is in my code file)*

I find some similar information from Stackoverflow[5] After understanding its meaning, I notice the type is different, so I make a simple help function to transfer the type and get it.

13. solve *(pass the test)*

In COMP1100 Study Event, Jay (Sorry, I forgot to ask his uniID) tells me the meaning of *backtracking*, he hints me to divide into four different parts and advices me to transfer to *Maybe Sudoku* at first, so actually I make a bridge to transfer the solve type from *String* to *Maybe*, in fact, I write two help function to transfer the type from String to Sudoku to *Maybe Sudoku* to [String], and the idea is just keeping trying from 1 to 9 over and over again.

14. solveNew *(seems not very efficient)*

Propagate is similar like a *tree* in week 8, Eriuo provides one idea about how to use it in his github,[6] his working shows me just directly transfer from Sudoku to Sudoku is not very efficient, I follow his instruction and transfer to *Maybe Sudoku* at first and then make some adjustment for some functions above. Propagate is a local consistence[7], for example, given a variable V with a domain of {1,2,3,4} and a constraint V≤3, node consistency would restrict the domain to {1,2,3} and the constraint could then be discarded. This pre-processing step simplifies later stages. I have to admit my whole

[4] https://cs.anu.edu.au/courses/comp1100/assignments/03/

[5] https://stackoverflow.com/questions/41126330/writing-sudoku-in-haskell-find-possible-candidates-for-a-cell

[6] https://github.com/Freezard/haskell-sudoku/blob/master/Sudoku.hs

[7] https://en.wikipedia.org/wiki/Local_consistency

framework for my solveNew function is based on him. In spite of in his working, some expression really confuses me, but I understand what he wants to do every step, so I follow his instruction, do some revise and make some helper function to fit the type I want, after several modifications, I get my solveNew function.

15. result

- solve *easy.txt* result:

```
real     0m47.212s
user     0m46.940s
sys      0m0.196s
```

- solveNew *easy.txt* result:

```
real    0m38.540s
user     0m38.348s
sys      0m0.180s
```

Conclusion

Compared with two different solutions, it is obvious that *backtracking* is a very basic solution without any technical skills, it runs very slow even if for *easy.txt*. Meanwhile, I think maybe I didn't use "propagate" correctly because I just solve the *easy* file a little fast and still can't solve the *hard* file quickly, I just follow others' idea and write my function based on his thinking. I still have a lot of places where are need to improve. The 20 warnings also show my code is not quite efficient.

Reference

https://stackoverflow.com/questions/41512822/error-when-changing-function-name-to-list-haskell

https://stackoverflow.com/questions/41126330/writing-sudoku-in-haskell-find-possible-candidates-for-a-cell

https://stackoverflow.com/questions/31360775/find-9-3x3-blocks-of-sudoku-board-haskell

https://cs.anu.edu.au/courses/comp1100/assignments/03/

https://stackoverflow.com/questions/41126330/writing-sudoku-in-haskell-find-possible-candidates-for-a-cell

https://en.wikipedia.org/wiki/Local_consistency

https://github.com/Freezard/haskell-sudoku/blob/master/Sudoku.hs

tutor: Yiping Su

help partners: Yian Hu (u5842618) , Shukai Zhang (u5686922)