SE-318 August 2014 Assignment 1

Set: Wednesday 3 September 2014

Due: Before Lecture Class (8.55am)Tuesday 9 September 2014

Marks: 15% of SE-318 total

Problem statement: write a Haskell function "successors" to compute the successors of a named member in a European-style Royal Family so that "successors N D" gives the list of names of the living persons in succession to the person named N in the dynasty D, using the rules (1) of male primogeniture that applied until recently, and (2) that abdication means that the person who has abdicated and all their descendants are omitted from the line of succession (as if the abdicated person had never lived).

Supplementary Information

The male primogeniture rule operates as follows:

- the successors of a person consist first of the combined successors of each of their offspring in order from oldest to youngest, but where male offspring are considered before female offspring;
- after that comes the combined successors of the person's following siblings (again according to the priority of age, but with males considered before females);
- after that comes the combined successors of the siblings following the person's parent, then grandparent, etc.
- dead persons are referenced in calculating priorities, but don't appear in the resulting line of succession.

Thus, to find successors after a person named N in some dynasty:

1. calculate the complete line of successors starting with the head of the dynasty;
2. then give the names of the living persons in that line after N

Hints and Instructions for Solution

In your solution, use the following types:

```
-- a person is Male or Female, Abdicated, Dead or Alive, and has a
name
data Sex = Male | Female deriving (Eq, Show)
data Status = Alive | Dead | Abdicated deriving (Eq, Show)
data Person = Person Sex Status String deriving (Eq, Show)
-- a Dynasty is headed by a Person and indicates the descendants
-- oldest first; a Dull Person doesn't have any recorded descendants
data Dynasty =
    Descend Person [Dynasty] | Dull Person deriving (Eq, Show)
```

In your solution, define and use the following functions:

```
successors :: String -> Dynasty -> [String]
successors name dynasty =  as specified above

linefrom :: Dynasty -> [Person]
linefrom dy =  the list of all Persons in order of succession in Dynasty dy

reorder :: Dynasty -> Dynasty
reorder dy = arrange for any immediate descendants of dy with males before females

sortds :: [Dynasty] -> [Dynasty]
sortds dys  = sort dys so that males precede females, not changing the order within each sex

insertd :: Dynasty -> [Dynasty] -> [Dynasty]
insertd dy dys  = insert dy into sorted position in dys (see sortds above)

aliveafter :: String -> [Person] -> [String]
aliveafter name ps =  the names of the Alive persons in ps occurring after name

alivein :: [Person] -> [String]
alivein ps =  the names of the Alive persons in ps
```

Example Dynasty

```
exdyn =
    Descend (Person Male Dead "George5")
    [
        Descend (Person Male Abdicated "Edward8") [],
        Descend (Person Male Dead "George6")
        [
            Descend (Person Female Alive "Elizabeth2")
            [
                Descend (Person Male Alive "Charles")
                [
                    Descend (Person Male Alive "William")
                    [
                        Descend (Person Male Alive "George") []
                    ],
                    Descend (Person Male Alive "Harry") []
                ],
                Descend (Person Female Alive "Anne")
                [
                    Descend (Person Male Alive "Peter")
                    [
                        Dull (Person Female Alive "Savannah"),
                        Dull (Person Female Alive "Isla")
                    ],
                    Dull (Person Female Alive "Zarah")
                ],
                Descend (Person Male Alive "Andrew")
                [
                    Dull (Person Female Alive "Beatrice"),
                    Dull (Person Female Alive "Eugenie")
                ],
                Descend (Person Male Alive "Edward")
                [
                    Dull (Person Female Alive "Louise"),
                    Dull (Person Male Alive "James")
                ]
            ],
            Descend (Person Female Dead "Margaret")
            [
                Dull (Person Male Alive "David"),
                Dull (Person Female Alive "Sarah")
            ]
        ],
        Dull (Person Female Dead "Mary"),
        Dull (Person Male Dead "Henry"),
        Dull (Person Male Dead "George"),
        Dull (Person Male Dead "John")
    ]
```

Correct Solutions based on Example

```
successors "Beatrice" exdyn
➔
["Eugenie","Edward","James","Louise","Anne","Peter","Savannah",
"Isla","Zarah","David","Sarah"]


linefrom exdyn
➔
[Person Male Dead "George5",Person Male Dead "George6",
Person Female Alive "Elizabeth2",Person Male Alive "Charles",
Person Male Alive "William",Person Male Alive "George",
Person Male Alive "Harry",Person Male Alive "Andrew",
Person Female Alive "Beatrice",Person Female Alive "Eugenie",
Person Male Alive "Edward",Person Male Alive "James",
Person Female Alive "Louise",Person Female Alive "Anne",
Person Male Alive "Peter",Person Female Alive "Savannah",
Person Female Alive "Isla",Person Female Alive "Zarah",
Person Female Dead "Margaret",Person Male Alive "David",
Person Female Alive "Sarah",Person Male Dead "Henry",
Person Male Dead "George",Person Male Dead "John",
Person Female Dead "Mary"]

reorder exdyn
➔
Descend (Person Male Dead "George5")
[
     Descend (Person Male Abdicated "Edward8") [],
     Descend (Person Male Dead "George6")
     [
          Descend (Person Female Alive "Elizabeth2")
          [
               Descend (Person Male Alive "Charles")
               [
                    Descend (Person Male Alive "William")
                    [
                         Descend (Person Male Alive "George") []
                    ],
                    Descend (Person Male Alive "Harry") []
               ],
               Descend (Person Female Alive "Anne")
               [
                    Descend (Person Male Alive "Peter")
                    [
                         Dull (Person Female Alive "Savannah"),
                         Dull (Person Female Alive "Isla")
                    ],
                    Dull (Person Female Alive "Zarah")
               ],
               Descend (Person Male Alive "Andrew")
```

```
                [
                        Dull (Person Female Alive "Beatrice"),
                        Dull (Person Female Alive "Eugenie")
                ],
                Descend (Person Male Alive "Edward")
                [
                        Dull (Person Female Alive "Louise"),
                        Dull (Person Male Alive "James")
                ]
        ],
        Descend (Person Female Dead "Margaret")
        [
                Dull (Person Male Alive "David"),
                Dull (Person Female Alive "Sarah")
        ]
    ],
    Dull (Person Male Dead "Henry"),
    Dull (Person Male Dead "George"),
    Dull (Person Male Dead "John"),
    Dull (Person Female Dead "Mary")
]
```

where exdyn is as above. (Newlines and indentation have been added by me to aid formatting in the above outputs, but you should not put these in your answer.)

Assessment Criteria

Presentation *The program should be laid-out in a readable, appropriately indented form, with sensible choice of names for functions and parameters (other than as specified above). Comments should be used to describe the behaviour of any other functions you define, and also where needed to describe any code that needs further explanation. Sensible reuse of Haskell Prelude functions instead of own written code is preferred, as long as the required functions above are all used.*	9 marks
Correctness *The program gives the correct output for the Example above, for operations* *1. linefrom exdyn* *2. successors "Beatrice" exdyn* *3. reorder exdyn*	6 marks
Total	**15 marks**

Note that the assessment is cumulative. That is:

- if the syntax is invalid, it will not be possible to get any marks for Presentation or Correctness
- marks for presentation will be proportional to the extent to which the program works correctly i.e. multiplied by the mark for Correctness divided by 5.

Submission Intructions

You should submit Haskell source code plus evidence of testing as required above.

Send one (1) email to **136619909@qq.com** by 8.55am on the due date in the precise format as follows:

- subject line: "SE-318 assignment 1" followed by your name and SYSU student number
- no body, but attachments as follows:
    1. a file named "a1.hs" – your source code, with definitions in order (from top to bottom)
        - data types Sex, Status, Person and Dynasty as above
        - functions
            - successors
            - linefrom
            - reorder
            - sortds
            - insertd
            - aliveafter
            - alivein

          Any other functions you think you need to define should be defined immediately after the function that first uses them.

          Leave at least one blank line between every function definition.
        - After the last function definition include the definition of exdyn as above so that you can use the name "exdyn" in your testing to refer to the test graph.
    2. a file named "a1.ss", being a screenshot of your testing of

```
linefrom exdyn

successors "Beatrice" exdyn

reorder exdyn
```

No submissions received after 8.55am on Tuesday 9 September will be assessed, and instead will be given a mark of 0/15.

Paul Bailes, 3.9.2014