German University in Cairo
Media Engineering and Technology
Dr. Nada Sharaf

Concepts of Programming Languages, Spring term 2019
Project Description
"Recommender System"

Deadline: 17.04.2019

The project will put your knowledge in Haskell to the test. Before proceeding, make sure that you read each section carefully. Enjoy.

Project Description

Almost all the applications we use today rely extensively on recommender systems to build user profiles and accordingly recommend items to buy/watch/read that matches their users' interests.

In this project, you are to build a recommender system that uses the users' purchases history to:

- Recommend an item to the user based on their previous purchases if the current cart is empty
- If there are items already added to the users cart, recommend an item to the user based on their previous purchases and the current items in the cart
- Another option is to recommend an item to the user based on the intersection between the items they previously purchased and other users' purchases

You are given the below toy dataset where `users` is a list of the users, `items` is a list of the items that the users can buy and `purchases` is a list mapping each user to a list of all his/her previous shopping carts.

```
users = ["user1", "user2", "user3", "user4"]
items = ["item1", "item2", "item3", "item4", "item5", "item6"]
purchasesHistory =  [
                        ("user1", [["item1", "item2", "item3"], ["item1", "
                            item2", "item4"]]),
                        ("user2", [["item2", "item5"], ["item4", "item5"]])
                            ,
                        ("user3", [["item3", "item2"]]),
                        ("user4", [])
                    ]
```

You could find the below function `randomZeroToX` useful which generates random numbers in the range from zero to X.

```
import System.Random
import System.IO.Unsafe

randomZeroToX :: Int -> Int
randomZeroToX x= unsafePerformIO (getStdRandom (randomR (0, x)))
```

Functions

Te rest of the description will show the functions that should be implemented.

recommend

`recommend :: String -> [String] -> String`

The function `recommend` takes a user and their cart and recommend to the user an item accordingly. The recommendation is always a random choice between

- the (non-empty) recommendation based on the cart and the previous purchases
- the (non-empty) recommendation based on the intersection between the items the user previously purchased and other users' purchases

If the two recommendations are empty, a random item is chosen.

Examples:

```
> recommend "user4" []
"item2"

> recommend "user1" []
"item4"

> recommend "user1" ["item1", "item2"]
"item3"
```

freqListItems

`freqListItems:: String -> [(String, Int)]` `freqListItems` takes a user and returns the frequency list of previously purchased items

```
> freqListItems "user1"
[("item2",4),("item3",2),("item4",2),("item1",4)]
```

freqListCart

`freqListCart:: String ->[String] -> [(String, Int)]` `freqListCart` takes a user and their cart and returns the frequency list of items in the users cart with respect to the previously purchased items

```
> freqListCart "user1" ["item2", "item4", "item6"]
[("item2",1),("item3",1),("item4",1),("item1",3)]
```

freqListCartAndItems

`freqListCartAndItems:: String -> [String] -> [(String, Int)]` `freqListCartAndItems` takes a user and their cart and returns the combinations of the frequency lists of

- items in the users cart with respect to the previously purchased items
- previously purchased items

```
> freqListCartAndItems "user1" ["item2", "item4", "item6"]
[("item2",5),("item3",3),("item4",3),("item1",7)]
```

recommendEmptyCart

```
recommendEmptyCart :: String -> String
```

When the users cart is empty, the function `recommendEmptyCart` recommends an item to the user based on the previously purchased items only. If the user's list of previously purchased items is empty, nothing is recommended.

Examples:

```
> recommendEmptyCart "user1"
"item2"

> recommendEmptyCart "user2"
"item5"

> recommendEmptyCart "user3"
"item2"

> recommendEmptyCart "user4"
[]
```

recommendBasedOnItemsInCart

```
recommendBasedOnItemsInCart :: String -> [String] -> String
```

The function `recommendBasedOnItems` is used to recommend an item to the user based on the items currently in the user's cart and the previously purchased items. If the user's list of previously purchased items is empty, nothing is recommended.

Examples:

```
> recommendBasedOnItemsInCart "user1" ["item5"]
"item1"

> recommendBasedOnItemsInCart "user1" ["item1"]
"item2"

> recommendBasedOnItemsInCart "user1" []
"item1"
```

freqListUsers

`freqListUsers:: String - -> [(String, Int)]` `freqListUsers` takes a user and returns the frequency lists of the intersection between the items purchased by the current user and other users

```
> freqListUsers "user1"
[("item3",3),("item1",6),("item4",2),("item2",3), ("item5", 2)]
```

recommendBasedOnUsers

```
recommendBasedOnUsers :: String -> String
```

The function `recommendBasedOnUsers` is used to recommend an item to the user based on the intersection between the items purchased by the current user and other users. If the intersection is empty, nothing is recommended.

Examples:

```
> recommendBasedOnUsers "user1"
"item3"

> recommendBasedOnUsers "user4"
""
```

getAllUsersStats

```
getAllUsersStats :: [(String, [[String]])] -> [(String, [(String, [(String, Int)])])]
```

The function `getAllUsersStats` calculates the frequency of items each user purchased.

Examples:

```
getAllUsersStats purchasesHistory

[
    ("user1",[("item1",[("item2",2),("item3",1),("item4",1)]),("item2",[("
        item1",2),("item3",1),("item4",1)]),("item3",[("item1",1),("item2
        ",1)]),("item4",[("item1",1),("item2",1)]),("item5",[]),("item6",[])
        ]),
    ("user2",[("item1",[]),("item2",[("item5",1)]),("item3",[]),("item4
        ",[("item5",1)]),("item5",[("item2",1),("item4",1)]),("item6",[])]),
    ("user3",[("item1",[]),("item2",[("item3",1)]),("item3",[("item2",1)])
        ,("item4",[]),("item5",[]),("item6",[])]),
    ("user4",[("item1",[]),("item2",[]),("item3",[]),("item4",[]),("item5
        ",[]),("item6",[])])
]
```

purchasesIntersection

```
purchasesIntersection :: Eq a => [(a, [(a, Int)])] -> [(a, [(a, [(a, Int)])
    ])] -> [[(a, [(a, Int)])]]
```

The function `purchasesIntersection` gets the intersection between the current user's and the other users' items statistics (frequencies).

Examples:

The below example finds the intersection between the frequency list of `user1` and the other users.

```
> purchasesIntersection ([("item1",[("item2",2),("item3",1),("item4",1)])
    ,("item2",[("item1",2),("item3",1),("item4",1)]),("item3",[("item1",1)
    ,("item2",1)]),("item4",[("item1",1),("item2",1)]),("item5",[]),("item6
    ",[])])
([("user2",[("item1",[]),("item2",[("item5",1)]),("item3",[]),("item4",[("
    item5",1)]),("item5",[("item2",1),("item4",1)]),("item6",[])]),("user3
    ",[("item1",[]),("item2",[("item3",1)]),("item3",[("item2",1)]),("item4
    ",[]),("item5",[]),("item6",[])]),("user4",[("item1",[]),("item2",[]),("
    item3",[]),("item4",[]),("item5",[]),("item6",[])])]

[
    [
        ("item2",[("item5",1),("item1",2),("item3",1),("item4",1)]),
        ("item4",[("item5",1),("item1",1),("item2",1)])
    ],
    [
        ("item2",[("item3",2),("item1",2),("item4",1)]),
        ("item3",[("item2",2),("item1",1)])
    ],
    []
]
```

createEmptyFreqList

```
createEmptyFreqList :: [a] -> [(a, [b])]
```

Initializes an empty frequency list that maps each item to a list that will contain all the items that were bought with this item along with their frequencies.

Examples:

```
> createEmptyFreqList items

[("item1",[]),("item2",[]),("item3",[]),("item4",[]),("item5",[]),("item6
    ",[])]
```