**Documentation of FurnishRoom Haskell Project**

**findFurnitureUpdate** is a predicate which takes an element c ("right" or "below") representing the position where the furniture b was found with respect to the examined furniture a and updates the statistics of the fourth input representing the so far conducted statslist according to c, in case c is "right" the appropriate value in the first list of the statslist is incremented (using findhelper) and if it is "below" the appropriate value in the second one is incremented (using findhelper), if found. Otherwise, it is inserted as a new element at the end of the list of either of them.

**findhelper** is a predicate that takes an object whose count in the statslist should be incremented but this predicate takes the exact list that should be traversed. If it found the element whose first term matches the string b, it increments the value of the integer and passes the resulting list to my\_sort to take care of keeping the order of the list (decreasing order: most to least frequent). If it didn't match this term, it recurses and sorts the current term with the resulting ordered list from the recursive call.

**my\_sort** takes care of the appropriate ordering of the statslist by keeping the most frequent elements before the rest.

**getPossibleNeighbour**:-

* It gets the total number of elements in both lists with their frequencies by the higher order function foldr.
* This number is then subtracted by 1 and sent to the randomZeroToX to get a random number to select the object accordingly
* Then the function getObject is called by that integer and then two lists the left and the below to get the object.

**getObject**:-

* It takes the integer that returned from the random function and then goes throught the left and right lists to get the element corresponding to it.
* First it traverses the left list and for each element if its frequency is higher than the remaining integer then it is the required element
* If the first list is empty then the required element is in the below one and the same happens on it.

**getFurnStats**:-

* It gets the statsList.
* Calls the getFurnStatsHelper with the element name and the statsList to get the list of right and the list of below element.

**getFurnStatsHelper**:-

* It traverses each pair of the statslist and returns the one corresponding to each element.

**foldHelper**

This is the function used by the foldr in the getPossibleNeighbour function. It takes as an input information about a certain element in the form (Name, Position, Frequency) and adds the frequency to the second input

**findRight** List:-

A function that that takes a list of lists representing the room. It outputs each pair of consecutive elements in each row in the following format (p1, p2,”right”)

**findUP\_helper** L1 L2:-

It is a helper function of the function findUP which takes two lists as an input. It outputs a list of elements in form (H1, H2, “below”) where H1 and H2 are the successive heads of the two lists. E.g. [H1:[H1: []]] & [H2:[H2:[]]] would be outputted as [(H1, H2, “below”), (H1,H2, “below”)]

**findUP** [[a]]

It is a function which takes as an input a list of lists representing a room and passes the first two to the findhelper function in order to generate a list of the “below” relationship between the first two rows of furniture. Afterwards it recursively calls itself with the second of these two lists in order to repeat the same process. The base case would be a list of one list, which would return this same list without passing it on to the findHelper function.

**feed** ((x,y,status):xs) list\_so\_far = feed xs (findFurnitureUpdate x y status list\_so\_far)

It is a helper function which takes as an input a list of elements of format [(Element1, Element2, Relationship)] and a list\_so\_far (a minor statsList) and recursively calls itself after it merges the head of the first list to list\_so\_far, as a second parameter as well as the rest of the first list as the first parameter. At the end when list1 becomes empty it returns the second list

**genR**

It is a helper function of the generate function which takes as an element two lists a room (2D array) and a list\_so\_far. Afterwards it “feeds” using the feed function a generated list of [(E1, E2, “right”)] into the second list

**generate** room list:-

generate is a function that takes as parameters a room in the form of list of lists and as well as a list that takes the design of statsList. It has the task of first passing on the room into the findUP function to generate a list of the elements of the room that satisfy the following format [(Elem1, Elem2, “below”)] (For simplicity let’s call it List A). Afterwards it generates a partially-finished statsList that consists only of the “right” relationship of the room and the original second parameter’s list using the “genR” function mentioned above. Finally it feeds the List A into that partially-finished statsList to produce the final statsList consisting of the statistics of the “room” and those of the “list”.

**statsList**

It is the paramter less function which has the task of generating a statistic list of the three rooms of the “training mode”. When this function is called it passes on “training” which is a list of the three rooms as well as an empty list to the statsHelper.

**statsHelper**

It is a two parameters function which takes first a list of rooms as input and initially an empty list. It then recursively calls itself until it goes through each room of the list. Then and only then it returns the second parameter which then would be the completely formed statsList. During each recursive call it generates a partially complete statsList using a room and then passes on this same list to the next recursive call.

Furnishing the room:

The design of this function is fairly straightforward. We first generate a 1D version of the room such that instead of taking this format [[Row1], [Row2], and [Row3]] it takes: [Row1, Row2, Row3]. After this version is generated it gets split using the my\_split method which divides them according to the required size (Both of them are passed as parameters to this method and its third parameter once base case is reached is the final format of the “room”).

The main helper functions used are:

**furnishRoom** which given the first furniture piece and the required size of the room it outputs a 2D array representation of a room. Its functionality is described above

**findElem**

It is pivotal function which takes as input a count of the element in my room so far as well the size (original one given to the furnishRoom) and a partially formed room. It generates the 1D version of the room and makes sure that each added element satisfies the constraints which they will have to follow when later on split into rows.

**locate**:- Takes as parameters a required position and a list and returns the element of the index equivalent to that position.

**getRightStats** Returns only the right statistics of a certain furniture X which it receives as an input from the getFurnStats method

**getBelowStats** Similar to that of the previous function however it works instead for the below statistics of X