Lavish Gulati - 170101082

CS431 - Programming Languages Lab - Assignment 3

3. House Planner

a. **Algorithm**

i. Compute all possible dimensions for each component given dimension range (a_1, b_1) to (a_2, b_2) by traversing over all values (a, b) s.t. $a_1 <= a <= a_2$ and $b_1 <= b <= b_2$
ii. Compute numbers of all components given input bedroom and hall numbers
iii. Get combinations of (bedroom, hall, kitchen, bathroom, garden, balcony) by concatenating each dimension tuple of all components
iv. Filter out all (bedroom, hall, kitchen, bathroom, garden, balcony) values in which total area >= max area, dimension of kitchen is greater than hall or bedroom, and dimension of bathroom is greater than kitchen
v. Remove all redundant tuples with the same area (as it does not affect final answer)
vi. Find maximum feasible area of all combination tuples
vii. Find (bedroom, hall, kitchen, bathroom, garden, balcony) tuple with area = max feasible area
viii. Print output

b. **Functions used**

i. possibleAreasHelper
ii. possibleAreas
iii. combinations1
iv. combinations2
v. combinations3
vi. combinations4
vii. combinations5
viii. redundant3
ix. redundant4
x. redundant5
xi. maximumArea
xii. design

c. **Are all those pure?**

design is an impure function because of the use of putStrLn, which is an impure function.

d. **If not, why?**

putStrLn is an I/O function as it involves printing to stdio (or terminal). This is because putStrLn causes a change in the program's state (which is generally caused by an expression or a function call, or modification of a global variable in a procedure or function) due to the change in the status of I/O. The presence of the IO type constructor means a function is impure, just as the absence of the const keyword in C/C++ means data might be modified.

Short Notes

1. **Lazy evaluation feature**
   Yes, lazy evaluation feature of Haskell can be exploited for better performance in the solutions to the assignments, because it helps in handling lists of very big size as they are evaluated only when needed. In question 3, the computation of possible dimensions is not executed when calling the function but when it is needed. Hence, lazy evaluation saves computation time and memory.

2. **Imperative language**
   Haskell has a major advantage of the lack of side effect. Lack of side effect means that variables are not changed once declared, and hence we can use parallel processing to compute such variables as they will not be changed at any point of execution, so we need not worry about consistency. Because of this reason, there won't be many critical sections and hence we can use parallel processing, which again saves computation time.