Assignment 03 Functional Programming in Haskell

By Shivam Bansal, Roll No 170101063

Pseudo code

```
1. possible_bed_sizes = [list of all possible sizes
      of bedroom from
      largest to smallest]
.
.
.
similarly for all rooms

2. get number of kitchen, bathrooms, balcony, garden
  from inputs and given constraints

3. iterate dimensions of all rooms over all the possible
  sizes generated in step 1

4. calculate space used after every dimension is selected
5. eliminate if space > total space available
6. Take head of resulting list to get most optimum solution
7. print it out
```

Number of functions

I used 4 functions —

- is_kitchen_size_valid — given kitchen, hall, bedroom dimensions check if kitchen dimensions are valid
- is_bath_size_valid — given bathroom, kitchen dimensions check if they are valid
- calc_space — given room counts and room dimensions, calculate used space
- plan — main function to do floor planning

Are they pure

Yes, all functions are pure.

There is no global data that is being mutated in any function.

And none of the functions have any side effects.

Lazy evaluation

Yes, I have used lazy evaluation to speed up my programme.

List comprehension that returns list of all possible solutions is computed lazily.

We only need most optimum solution so we take it's first element.

Also, because of laziness, recursion inside the list comprehension is much faster.

Comparing with imperative languages

- Strong type inference ensures there are no errors at compile time
- Because everything is lazily evaluated, it is easy to work with large lists. It is easier to initialize them as well
- Since functions have no side effects, it is easier to refactor code fearlessly of breaking something else somewhere else.