Programming Languages Lab

Assignment 3 - Functional Programming

Sahithya Vemuri (170101077)

House Planner:

Algorithm:

- Get all the possible dimensions for bedroom(s), hall(s), kitchen(s), bathroom(s), balcony and garden, given the smallest and largest dimension.
- Find the cartesian product of dimensions pairwise as a list and then filter the list with all the conditions mentioned in the question.
- Remove all the extra tuples that will add up to give the same total areas for computational efficiency.
- After getting all the tuples without any duplicates calculate the maximum area that is possible from these tuples.
- Get the tuple with the maximum area which is calculated above by filtering the final list and then print the final result.

How many functions did you use?

- I've used 15 functions in total which are given below:
    - design
    - getDim
    - getDim2
    - rmDup2
    - rmDup3
    - rmDup4
    - rmDup5
    - rmDup6
    - tempDup2
    - tempDup3
    - tempDup4

    - tempDup5
    - tempDup6
    - getMaxArea
    - printResult

Are all those pure?

- For design function, input is taken from the keyboard so it changes the state of the function and hence it is not a pure function. All the remaining functions are pure functions.

If not, why? (that means, why the problem can't be solved with pure functions only).

- For these kinds of problems we need to take input from the user and then find the desired output which can be done by using only impure functions. Therefore such problems cannot be solved with the use of only pure functions.

Short Notes

Do you think the lazy evaluation feature of Haskell can be exploited for better performance in the solutions to the assignments? If so, which solution(s) and how?

- Yes I think that the lazy evaluation feature of Haskell can be exploited for better performance in the solutions to the assignments. This feature of Haskell can mainly help in handling lists of large sizes.
- In problem 3 House Planner, we are using lists of very large size and these are being evaluated with lazy computations when needed.

We can solve the problems using any imperative language as well. Do you find any advantage of using Haskell for these problems (w.r.t the property of lack of side effect)? If your answer is no, elaborate on why not?

- Haskell provides a lot of features such as "Lazy Evaluation", "No Side Effect", etc, like many other functional languages.
- Values of variables do not change in No Side Effect, hence it helps in debugging and making the code modular.
- Since the variable's value does not change we need not consider global variables.

- As the variable’s value does not change, functional programming languages can help us exploit parallelism.
- There won’t be any critical section related to assignment operations of variables.
- Lazy Computation saves us some time that is helps to avoid unnecessary computations and will compute a value as and when required only.
- This feature helps in saving a lot of computation time by making the program efficient.