CS431: Programming Language Lab
Assignment 3
Haskell

Roll no. - 170101022

Name – Devansh Gupta

Question1

- How many functions did you use?

  I had used following functions: -
  - performSubstraction
  - performAddition
  - performUnion
  - performNullCheck
  - performIntersection
  - removeDuplicates
  - union
  - inter
  - add
  - sub
- Are all those pure?
  - All functions are pure if we assume input hardcoded.

Question2

- How many functions did you use?

  I had used following functions
  - nextMatch
  - fixture
  - display_fixture
  - zipper
  - teams_list
- Are all those pure?

No, nextMatch and fixture are not pure while display_fixture, zipper and teams_list is pure.

- If not, why?

  Input from user can be taken by using impure functions only therefore this problem can't be solved without impure functions.

Question3

- Write the algorithm. (Pseudo code)
  - find the all possible dimensions of all the components given the maximum and minimum size.
  - We eliminate those combinations of components which don't able to satisfy given conditions.
  - Then we find area for all the possible component combinations remaining.
  - Then we select the combination with maximum area.
- How many functions did you use?

  I had used following functions:-
  - findAllPossibleDim
  - eliminateRepeated
  - cartProd
  - design
  - findMaximumArea
- Are all those pure?

  No, design is not pure while all the remaining that is findAllPossibleDim, eliminateRepeated, cartProd and findMaximumArea are pure.
- If not, why?

  Input from user can be taken by using impure functions only therefore this problem can't be solved without impure functions. In this case input is taken from design function.

Short Notes: -

a) 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?

It allows the language runtime to discard sub-expressions that are not directly linked to the final result of the expression. It reduces the time complexity of an algorithm by discarding the temporary computations and conditionals. It allows the programmer to access components of data structures out-of-order after initializing them, as long as they are free from any circular dependencies. Most importantly this feature can help in handling data structure of very big size. In the house planning question we had exploited the above features Haskell to solve our problem to solve our problem efficiently.

b) 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 provide features such as “No Side Effect” and “Lazy Evaluation”. In no Side Effect, value of variables is not changed and hence it helps in making our code modular and helps a lot in debugging. As values cannot be changed, we need not to consider about things like global variables. As values of variables is not changed, functional programming languages can help us exploit parallelism. There will not be any critical section involving assignment operations of variables. Lazy computation helps to save unnecessary computations and compute a value as and when needed. This helps in reducing the amortized time complexity of program, making the program efficient.