




NOVEMBER 19, 2018

ASSIGNMENT 3

FUNCTIONAL PROGRAMMING WITH HASKELL

GROUP24 SHIVAM GUPTA(150101068) PIYUSH JAIN(150101046)



Problem-1 (Palindrome Maker)

1. How many functions you used?

There are 3 functions in our code.

- Replace
- Pal
- main

2. Are all those pure?

Out of these 3 functions, 2 of them are pure i.e. Replace and Pal. In the main function, input is taken from keyboard and hence it changes state of function and hence It is not a pure function.

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

In any problem, we need to take input from user which can be done by using impure functions only. Hence such problems can't be solved with pure functions only.

4. How can you use impure functions in Haskell?

We have used impure functions for I/O operation.

Problem-2 (Free coupon Problem)

1. How many functions you used?

There are 2 functions in our code.

- Usetoken
- main

2. Are all those pure?

Out of these 2 functions one of them is pure i.e. Usetoken. In the main function, input is taken from keyboard and hence it changes state of function and hence it is not a pure function.

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

In any problem, we need to take input from user which can be done by using impure functions only. Hence such problems can't be solved with pure functions only.

Probelem-3 (Minimum Number of Moves)

1. In order to get the minimum number of the operations, which worker should be chosen and why?

In this ques, worker with the maximum salary should be chosen. When we increase salary of all workers except the worker with maximum salary, then we are decreasing difference in salary of workers and hence step by step we will find ans.

2. How many functions you used?

There are 2 functions in our code.

- `getAns`
- `main`

3. Are all those pure?

Out of these 2 functions one of them is pure i.e. `getAns`. In the main function, input is taken from keyboard and hence it changes state of function and hence it is not a pure function.

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

In any problem, we need to take input from user which can be done by using impure functions only. Hence such problems can't be solved with pure functions only.

Problem-4 (House Planner)

1. Write the algorithm (in pseudo-code) that you devised to solve the problem (you must not write the code for the algorithm).

Algorithm:

- **getAll**: Given smallest and largest dimension, get all possible dimensions for each of the six entities.
- **cartProd**: Find Cartesian Product of pair of dimensions pairwise as a list.
- **filter**: Filter this list with conditions given in question.
- **removeDuplicates**: Remove duplicate tuples which add up to give same areas. This will reduce unnecessary computations and make our program faster.
- **getAns**: Calculate max area possible from all the tuples generated finally.
- **filter**: Filter the final list to get the tuple with the above calculated maximum area.
- **print** the result.

2. How many functions you used?

We have used following functions in our code.

- getAll
- getAll2
- cartProd2
- cartProd3
- cartProd4
- cartProd5
- cartProd6
- removeDuplicates2
- remDups2
- removeDuplicates3
- remDups3
- removeDuplicates4
- remDups4
- removeDuplicates5
- remDups5
- removeDuplicates6
- remDups6
- getAns
- getMaxArea
- main

3. Are all those pure?

In the main function, input is taken from keyboard and hence it changes state of function and hence it is not a pure function. Except this main func, all other functions are pure.

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

In any problem, we need to take input from user which can be done by using impure functions only. Hence such problems can't be solved with pure functions only.

Common Questions

1. 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 lazy evaluation feature of Haskell is very good and can be exploited to solve many problems. This feature can help in handling lists of very big size.

In ques 4 we are dealing with lists of big size and these lists are evaluated with lazy computations as and when needed.

2. 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?

Like many other functional languages, 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 saving much computation time, thus making the program efficient.