Assignment Set 3

Functional Programming with Haskel

By: Saket Sanjay Agrawal (150101054)

Anirudh Sharma (150101006)

Problem-1 (Palindrome Maker):

Q1. How many functions you used?

3 functions are used. They are the following:

1. **absolute_difference_between_characters**: Calculate absolute difference between ASCII values of characters.
2. **find_min_operations**: Finding minimum operations to convert the given string to palindrome. First we find the absolute difference pairwise between two arrays namely x and y(reverse of x). Then we sum this resulting array of difference and finally we divide this by two.
3. **main**: Take input string, process it by calling suitable functions and then print the required value.

Some inbuilt functions are also used - abs(), ord(), zipWith(), sum(), putStrLn, getLine, print().

Q2. Are all those pure?

Functions 1 and 2 are pure. There is no external side effect in those functions.

Function 3 i.e main is impure function since it involves input-ouput

functions namely putStrLn, getLine, print.

Q3. If not, why? (that means, why the problem can't be solved with pure functions only).

The I/O operation are necessity for this problem. We need to take string as input in order to identify the palindrome. Since function with I/O operation is an impure function, so we can't solve the problem with all pure functions.

Q4. How can you use impure functions in Haskell?

Functions that do input output are impure functions. So, we can use this I/O functions in our Haskell program.

Problem-2 (Free coupon Problem):

Q1. How many functions you used?

3 functions are used. They are the following:

1. **find_coupons**: Function to find number of parathas which can be bought using coupons
2. **find_max_parathas**: Function to find number of parathas which can be bought with given amount and then from the coupons.
3. **main**: Take input string, process it by calling suitable functions and then print the required value.

Some inbuilt functions are also used - putStrLn, getLine, print().

Q2. Are all those pure?

Functions 1 and 2 are pure. There is no external side effect in those functions.

Function 3 i.e main is impure function since it involves input-ouput

operations.

Q3. If not, why? (that means, why the problem can't be solved with pure functions only).

The I/O operation are necessity for this problem. We need to take the price of paratha, exchange rate between coupons and paratha and total amount carried initially as input. Finally we need to output the max parathas that one can buy with given inputs. Since function with I/O operation is an impure function, so we can't solve the problem with all pure functions.

Probelem-3 (Minimum Number of Moves) :

Q1. In order to get the minimum number of the operations, which worker should be chosen and why?

To get minimum number of operations, we have to choose person with highest salary and increase salary of other people. As the only possible move is to increment salary, hence increasing salary of the people having low salary would take less step to reach common salary because one way or other lower salaries have to increased in order to have common salary and it would be wise to not increase the highest salary. Hence greedy paradigm would be to pick highest salary guy and increase salary of others.

Q2. How many functions you used?

3 functions are used. They are the following:

1. **process**: Functions that take input as workers salary and output the minimum operations required to equal the salaries.
2. **main**: Take input string, process it by calling suitable functions and then print the required value.

Some inbuilt functions are also used - sum, minimum, map, putStrLn,

getLine, readLn, print(), words.

Q3. Are all those pure?

Function 1 is pure. There is no external side effect in those functions.

Function 3 i.e main is impure function since it involves input-ouput operations.

Q4. If not, why? (that means, why the problem can't be solved with pure functions only).

The I/O operation are necessity for this problem. We need to take the number of workers and their initial salaries as input. Finally we need to output the minimum operations required to make their salaries equal. Since function with I/O operation is an impure function, so we can't solve the problem with all pure functions.

Problem-4 (House Planner)

Q1. Write the algorithm (in pseudo-code) that you devised to solve the problem (you must not write the code for the algorithm).

max_used_space = 0;

answer = [];

Quantities of each component is fixed by the input and constraints.

for each possible bedroom dimension (10 X 10) -> (15 X 15) {

for each possible hall dimension (15 X 10) -> (20 X 15) {

for each possible kitchen dimension (7 X 5) -> (15 X 13) {

for each possible bathroom dimension (4 X 5) -> (8 X 9) {

for each possible balcony dimension (5 X 5) -> (10 X 10) {

for each possible garden dimension (10 X 10) -> (20 X 20) {

-- now the dimension for each component is fixed.

Check if it satisfies all the given constraints: The dimension of a kitchen must not be larger than that of a hall or a bedroom. The dimension of a bathroom must not be larger than that of a kitchen.

If it satisfies then update the answer if the space used by this configuration is larger than max_used_space.

}}}}}}

print the answer if max_used_space is greater than 0 (it means atleast one configuration exist) else print not possible.

Q2. How many functions you used?

12 functions are used. They are the following:

1. **calc_bed_x**: Functions that explores possible dimensions of configuration of bedroom and returning most optimal one.
2. **calc_hall_x:** Functions that explores possible dimensions of configuration of hall and returning most optimal one.
3. **calc_kitchen_x:** Functions that explores possible dimensions of configuration of kitchen and returning most optimal one.
4. **calc_bathroom_x:** Functions that explores possible dimensions of configuration of bathroom and returning most optimal one.
5. **calc_balcony_x:** Functions that explores possible dimensions of configuration of balcony and returning most optimal one.
6. **calc_garden_x:** Functions that explores possible dimensions of

configuration of garden and returning most optimal one.

7. **final_check:** Checking that the given configuration(int local) satisfies all constraints.
8. **comp:** Compare two answers based on which has better space utilization
9. **calc_area:** Calculate area used for given configuration.
10. **showsArc:** Function used to print final answer.
11. **find_kitchen:** To derieve number of kitchen from number of bedrooms
12. **main**: Take input string, process it by calling suitable functions and then print the required value.

Some inbuilt functions are also used - take, drop, showString, shows, ceiling, fromIntegral, putStrLn, getLine, print.

Q3. Are all those pure?

Function 1-9 and 11 are pure. There is no external side effect in those functions.

Function 3 and 10 are impure functions since it involves input-ouput operations.

Q4. If not, why? (that means, why the problem can't be solved with pure functions only).

The I/O operation are necessity for this problem. We need to take the number of workers and their initial salaries as input. Finally we need to output the minimum operations required to make their salaries equal. Since function with I/O operation is an impure function, so we can't solve the problem with all pure functions. We could have avoided function 10

and print the final answer in main only.

In addition, write short notes on the following in the report.

Q1. 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?

Lazy evaluation is a method to evaluate a Haskell program. It means that expressions are not evaluated when they are bound to variables, but their evaluation is deferred until their results are needed by other computations. In consequence, arguments are not evaluated before they are passed to a function, but only when their values are actually used.

So, it saves the time for the expressions which are not used in the program.

It also helps in debugging.

No, in our assignment we havent used the lazy evaluation property of Haskell.

Q2. 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?

Some of the advantages of using functional programming are:

- Purity makes the job of understanding code easier.
- Testing is easier, and pure functions lend themselves well to techniques like property-based testing.

- Debugging is easier.
- Parallel/concurrent programming is easier.