COL100 Assignment 3

Due date: 5 January, 2021

General assignment guidelines

1. For each question, the algorithm must be defined mathematically in your written work, as well as implemented in SML code in your programming component.
2. Analysis of correctness and efficiency must be carried out for all nontrivial helper functions as well as for the main function. Any function that performs recursion counts as nontrivial, as does any function that calls a nontrivial function.
3. Big O notation should be as tight as possible. For example, reporting $O(n^2)$ complexity for an $O(n)$ algorithm could lose you marks, even though it is technically correct.
4. For the programming component of each question, the name and the type of the main function must be exactly as specified in the question. You can take help of the test cases provided to validate your code
5. Any function that has integer input and output must not perform any intermediate computation using reals.

1 Operations on very large numbers

You are given the task of performing some mathematical operations on very large numbers. However, you find that the numbers given are too large to be stored in variables like int (which has a limitation of max 1073741823). You then have a brilliant idea and decide to store these numbers in the form of base 10 lists of digits.
The lists are implemented in such a way that the least significant digit occupies the first position in the list and the most significant digit occupies the last digit.
So as an example, the number 123 would be represented as [3, 2, 1] in the list format and 912 would be [2, 1, 9]. On adding these two we should get [5, 3, 0, 1] which is 1035.

1. Define an algorithm for the following operations on large numbers: (2 marks)
   (a) Convert a given integer number into the list format.

(b) Convert a number in list format into its equivalent integer. (If the number in list format exceeds 10^9, then the function should just return 10^9).

(c) Add two numbers in list format.

(d) Compare two numbers in list format. The algorithm should return True if $l1 \leq l2$ and False otherwise.

2. Prove the correctness of your algorithms and analyze their time and space complexities. (2 marks)

3. Implement the four algorithms as four functions: (3 marks)

   (a) `LgintToInt` : `int list` $\longrightarrow$ `int`

   (b) `intToLgint` : `int` $\longrightarrow$ `int list`

   (c) `addLgint` : `int list` $*$ `int list` $\longrightarrow$ `int list`

   (d) `LgLesseq` : `int list` $*$ `int list` $\longrightarrow$ `bool`

Bonus: Design an algorithm for multiplication of two numbers in list format and implement your algorithm in the form of a function `multiplyLgint` : `int list` $*$ `int list` $\longrightarrow$ `int list`.

2 Quarterly Performance

You are given a list of tuples of employees in a start-up, where each tuple contains an employee's performance points for each quarter in the last year and their current salary ($salary = 100 * i$ for some $i \in \mathbb{N}$).
You are expected to find the average performance of each quarter for all the employees and if an employee's performance is above the average performance of that quarter provide them with a salary hike of 1% increment on base salary when at least 10% gain compared to average and in increments of 1% for every additional 10% difference between their points and average. For example, if their points are 42 and if the average for that quarter is

- >= 38.2, they get no hike from that quarter
- = 36, they get 1% hike
- = 30, they get 4% hike

The total salary hike for an employee will be done accordingly using all quarters and output the resultant salaries of all the employees.
For example, if the input is [(10, 20, 30, 40, 100000), (30, 30, 20, 50, 150000), (60, 10, 10, 50, 200000)], the output would be [105000, 157500, 216000]. Finally, calculate the overall percentage raise in salary budget the start-up has to handle solely considering salary hike. which would be 0.0633... for the above example

1. Design an algorithm to apply the above logic on a list of employees and argue on its correctness. You should use higher order list functions like map, filter and foldl/r as to the extent possible. (3 marks)
2. Analyse the time and space complexity of your algorithm. (1 marks)
3. Implement your algorithm containing functions `qPerformance : (int * int * int * int) list → int list` and `budgetRaise : (int * int * int * int * int) list → real`. (2 marks)

3 Lexicographic Permutations

Lexicographic ordering is what you may have already noticed in the ordering of words according to alphabet in the dictionary.
More formally, Given two lists $[a_1, b_1]$ and $[a_2, b_2]$, they are said to be $[a_1, b_l] < [a_2, b_2]$ in a lexicographical order iff either $a_1 < a_2$ or $a_1 = a_2$ and $b_1 < b_2$. This can be extended to lists of characters of arbitrary length by recursively applying this definition.
For example, $[a, b, c] < [a, c, b] < [c, a, b]$ if we are to take the English alphabet order for the lists of characters. For example for

1. Design an algorithm which enumerates all permutations of the given list of distinct characters in lexicographic order and prove its correctness. For example for input [a,b,c] the output will be [[a,b,c],[a,c,b],[b,a,c],[b,c,a],[c,a,b],[c,b,a]]. (3 marks).
2. Analyze the time and space complexities of your algorithm. (1 marks)
3. Implement your algorithm as a function `lexicographicPerm : 'a list → 'a list list` (3 marks)

Bonus: Implement a new function `lexicographicPermDup : 'a list → 'a list list` to handle the input containing duplicate characters. The output list should have no duplicate permutations.

4 Submission and other logistics

As usual, you should submit two files to the Moodle assignment page.

1. One file should be a PDF file containing all your written work, i.e. mathematical definitions of algorithms, correctness and efficiency analysis, etc. This can be handwritten with pen and paper and then scanned with your mobile, or it can be typed on your device using MS Word or LaTeX or any other software. The only requirement is that it should be clearly legible.

2. The other file should be an SML file containing all your code. Put your solutions to all four programming problems into a single file, along with any helper functions they require. We should be able to run any of the four required functions by typing in a function call at the bottom of the file.

The filenames of your submitted files should be exactly the same as your entry number. For example, if your entry number is 2017CS10389, you should name your SML file as `2017CS10389.sml` and your PDF file as `2017CS10389.pdf`. Your submission should consist of only these two files, uploaded individually. There should be no sub-folders or zip files in the submission on Moodle.

Failure to comply with these submission instructions will lead to a penalty.

Please ask any queries related to the assignment on the COL100 Piazza page (`https://piazza.com/iit_delhi/fall2020/col100`).