**(S2-20\_DSECLZG519) (Data structures and Algorithms Design)**

**Academic Year 2020-2021**

Software Design Documentation

**Assignment 1 PS5 – Assignment - Group 21**

**Overview:**

## This problem is about a giant company to keep a record of companies bought and sold by itself

**Given Problem Statement:**

You are working for a giant conglomerate. Increasing their presence and utilising free cash is by investing/buying and selling companies. Sometimes an acquired company could invest and buy another smaller company and sometimes could also sell a company

You must keep a record of companies bought and sold such that each node represents a company and the immediate child nodes are the companies purchased by the conglomerate division. Implement following functions

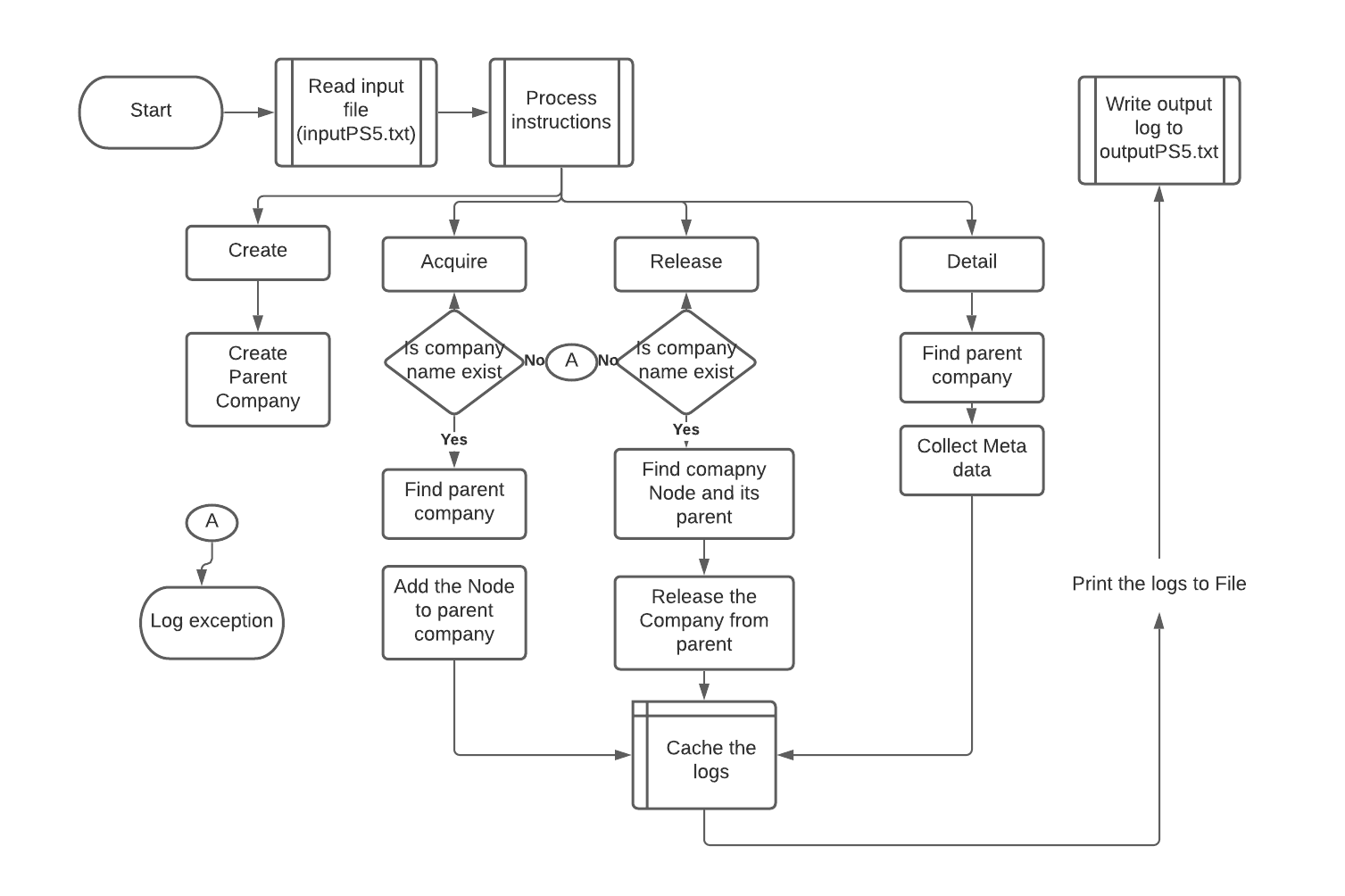
* 1. a. **detail**: Display the details of the mentioned Company. Who acquired the Company and what the said Company acquired other companies
  2. b. **acquire**: Adds a newly acquired company to the records.
  3. c. **release**: Release or sell off the mentioned Company

**Goal:**

The goal is to create a python program to read the list of commands and perform the same operations and keep a record of the functions in the LOG

**Flow Diagram:**

The following diagram shows the overall flow



**Design + Data Structures:**

We have used the following data structures to perform the expected operations

* GeneralTreeNode: The custom class consists of two members attributes
  + A string variable to hold the data item value
  + A list to save its children which is of GeneralTreeNode Type
* Company: Custom class consists of three-member attributes
  + ParentCompany: The root company and it is of datatype GeneralTreeNode
  + All\_company\_names\_cache: This is SET data type to hold the list of company names, and it gets updated whenever the parent company acquires or releases the companies. This is handy to validate the company names before acquisition or release by not traversing the entire tree to find the terms of the acquired companies
  + Trace\_log: List to hold the result of operations

This is the simplest data structure meeting all our requirements.

**Run Time Analysis:**

Here is the Run time analysis for each operation

The complexity of this program for any of the operations is O(n power 3)

**Alternate Modelling:**

**Find Node: This c**an be implemented using recursive logic, increasing the time complexity. The complexity can be n to the power n. Hence we used tree traversal logic to find the node with the order of complexity of n power3

**Algorithm:**

|  |
| --- |
| **Input:** File inputPS5.txt contains list of commands / operations.  E.g.  "Company: ce",  "No of operations: 6",  "DETAIL ce",  "ACQUIRED:aviation BY:ce",  "ACQUIRED:power BY:ce",  "RELEASE aviation ",  **Output:** File OutputPS5.txt lists the maximum bonus earned and the sequence of Jobs for every test case.  E.g.  'DETAIL:ce'  'Acquired companies:None'  'No of companies acquired:0'  'ACQUIRED SUCCESS:ce succesfully acquired aviation'  'ACQUIRED SUCCESS:ce succesfully acquired power'  'ACQUIRED SUCCESS:ce succesfully acquired healthcare'  'DETAIL:ce'  'Acquired companies:aviation, power, healthcare'  'No of companies acquired:3'  'RELEASED SUCCESS: released aviation successfully'  **Function:** The following algorithm gives a high-level overview of the overall program   * **Main**   + Reads the file contents and caches them into a commands list (String datatype)   + **ProcessInstructions:**     - Iterates through the command’s list and performs Create, Detail, acquire, release actions/functions based on the value command     - Create       * Create operation creates a new company if the command is Company. It is assumed the first command in the input file is Company     - Acquire       * It takes two parameters: parent company name and acquired company.       * If the acquired Company already exists in the set of already acquired companies, then the program throws an error       * Else, the program finds the parent company name through find\_company routine and gives the node       * It then adds the acquired company name to the children list fo the parent node       * Also it adds the company name to the cache       * It then track the details to the trace log     - Release       * It takes the company name to be released and searches for the Company in the cache. If it exists, then it proceeds else throws an exception       * If the company name exists, then the program uses the find\_company to search the parent node of the Company to be released       * One the parent node is found, the Company to be release gets removed from the children list       * It then updates the cache by removing the Company to be released       * It then track the details to the trace log     - Detail       * It takes the company name and finds the node using find company routine       * If then the takes names of immediate children’s companies       * It then adds the details to the trace log ie., number of immediate children and their names     - Find Company       * This is depth first algorithm to traverse the tree and compares the company name and once it is found it returns the node else None   + The previous step, process instructions, returns the Trace log in list datatype. The final step is to write the log into a file |
|  |