COL100 Assignment 12
2^{nd} Semester Semester : 2021-2022

Deadline: 11:59 pm, 23 June, 2022

General Instructions

You should attempt this assignment without taking help from your peers or referring to online resources except for documentation (we will perform a **plagiarism check** amongst all submissions). Any violation of above will be considered a breach of the honor code, and the consequences would range from **zero marks** in the assignment to a **disciplinary committee action**.

Submission Instructions

1. Please write efficient code that and uses minimum time to run. We will be giving 4 seconds of time for your code to produce output on autograder. Strict penalties for writing inefficient code.
2. **Your code must be in a file named `<EntryNo>-q.c`.** Example: if your Entry Number is `2018CS50402` then your file must be named `2018CS50402-q.py`. **Please ensure this point.**
3. You must submit the lab question mentioned in the assignment as "In Lab Component". The question evaluation for a lab will be done in the lab only and evaluations not done for the in-lab component for a student in his/her lab slot will be marked 0. It is your duty to get them evaluated.
4. **Submit your code in a `.zip` file named in the format `<EntryNo>.zip`. Make sure that when we run unzip `<EntryNo>.zip`, a folder `<EntryNo>` should be produced in the current working directory.** For eg. if your entry number is `2021CS5XXXX`, then your zip file would be `2021CS5XXXX.zip` and upon unzipping, it should produce a folder `2021CS5XXXX` containing file `<EntryNo>-q.c`. We have provided a zip that contain the skeleton code with this directory structure and similar naming convention. **Please ensure this point.**
5. Your submissions will be **auto-graded**. Make sure that your code follows the specifications (including directory structure, input/output, importing libraries, submission `.zip` file) of the assignment precisely.
6. You have to write all your code in C only.

Some Clarifications

1. Every problem description is followed by some examples showing how exactly input and output are expected. Please refer to them for more clarity. **Please ensure this point is followed.**
2. Write your code only in the skeleton code provided.
3. Do not change function names, argument order, and input-output statements , etc already in the skeleton code.
4. Do not comment the `main` function. Just implement the functions that you are asked to.
5. You can include Lists from previous assignment (Assignment 10), if needed. We are releasing the solutions, you can make the necessary changes i.e. create a header file and include it in the headers or maybe copy-paste the code. If you include the previous assignment code you must include the other files in the submission as well, or it would not run. You are not allowed to use any external header files apart from assignment 10 and the ones already in the driver code.
6. If you still have any more doubts, feel free to shoot them on Piazza.

1 Organisation Hierarchy

Note: Please read the description carefully for attempting the questions as there are some changes with respect to Assignment 11. This assignment is an extension from Assignment 11 and most keywords remain the same.

I am the CEO of a "Fortune 500" company and we have to create a software to manage all the employees. Unfortunately, I don't know how to code therefore I request you to help me with it. My organisation has many levels; with me at the top i.e. level 0. You did a great job last time and now I want to add some new functionality. Each employee (including me) may be in charge of at most two people. An employee of my organisation has an **Employee ID** (unique in the organisation and is a non-negative integer), **Employee Name** (It consists of one word the Surname or Last name), **Employee Salary** (This is a non-negative integer) and **two pointers** which points to the subordinates. For the sake of convention lets call them **subordinate one** and the **subordinate two** of the employee. Let us understand the terms of my company in detail.

We have a weird tradition of calling people by their last name however this causes some confusion. You will be helping us with this. Note that the name of two employees may be the same but their id will be unique.

1. **Employee:** This is a structure in C that contains **Employee ID, Employee Name**, **Employee Salary** and **two pointers** which points to the subordinates.
2. **CEO:** The CEO is the top most employee of the company that has no manager above them. There is only one CEO in my company.
3. **Edge:** Edge acts as a link between the manager and the subordinates.
4. **Intern:** An employee that has no subordinates is known as the Intern. It is the last employee of the company. There can be multiple interns in a company.
5. **Level:** Level of the employee is the distance from the CEO to that particular employee i.e. count of how many positions is an employee below the CEO in hierarchy of the company. The CEO is at level 0.
6. **Subordinate:** This is an employee that works **directly** under another employee.
7. **Team:** It is a sub-organisation inside the company. **The team with the CEO as head is the company itself**. Then we have two teams - one led by subordinate one and the other led by subordinate two. Each team can have at max two sub teams lead by the subordinates of the team head.
8. **Company:** The collection of all the employees working under the CEO. It is also called organisation. Both words are used interchangeably.

In this assignment you are given this structure of an organisation and you have to implement certain functions.

You are given a global pointer `CEO` that always points to the head of the Company and some helper function to create an employee and to print the employees. You are not allowed to change these helper

functions that are already there since any changes might interfere with the Auto-grader. Also, make sure that `CEO` points to the head of the company always. Not following this invariant might result in malfunctioning of the Auto-grader.

Some description of the helper functions provided to you:

1. `create_employee(int id, char* name, int salary)`: This function would create a new `Employee` with the `emp_id` attribute as `id`, the `emp_name` attribute as `name` and the `emp_salary` as the salary. The `subordinate one` and `subordinate two` pointers are set as `NULL`. This function returns a pointer to the employee. Note that the creation of the structure is done by us and **we will provide only non negative unique IDs**.

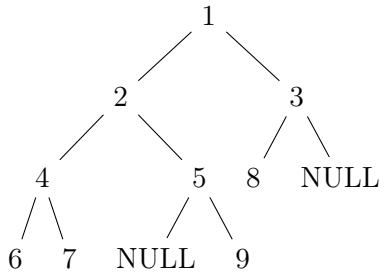
2. `create_company()`: This function would create a new `Company` from the input given in the `input.txt` and return a pointer to the CEO.

3. `print_company(struct Employee* head)`: This function would print the organisation at any moment for you starting from the head. You can pass another pointer to print the team with the given employee as the head.

You have to implement the functions mentioned below and are also provided with the `main` function. Again, as above, do not change the `main` function. You can change the inputs given to the program to check your code, you may also add print statements in your functions but make sure before submitting, you comment/remove all such print statements.

Hint: Please see the implementations of the Helper functions carefully to get a fair idea about how to implement the functions.

Please download the skeleton code from Moodle. Please read the comments of the Skeleton code for some clarity on the functions. You only need to fill the functions in the skeleton code and change nothing else. You can't change the parameters and the return types of the functions in the skeleton code. You just have to return the values in the functions and rest will be handled by the `main` given to you.



Consider the above company, the input file will contain the list of employee IDs, employee names and employee salaries together. Name and salary has not been included in the figure here to ensure readability, but after each id there is a string which corresponds to the name and a integer which corresponds to the salary. Therefore in the explanation below we have omitted name and salary.

The input format will be like `employeeId employeeName employeeSalary`. The only change between assignment 11 and this one is that instead of only id we also provide name and salary right after.

The input `1 2 4 6 -1 -1 7 -1 -1 5 -1 9 -1 -1 3 8 -1 -1 -1` , based on this list the `create_company()` will create the given company hierarchical structure. Here we have followed the convention that at first

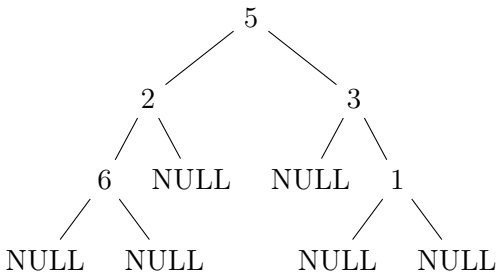
we have the employee id (and name and salary which is not included here for the sake of comprehensibility) of the CEO followed by the employee id of subordinate one. Then we will have the values assuming subordinate one to be at top. Then followed by ids assuming subordinate two to be on top. This is called a pre-order traversal and you can read about it more online here.

The actual input will be something like `1 Prakash 1000 2 Bansal 300 4 Jain 100 and so on` This order will always be unique for a company and we handle the creation of the same. Some explanations -

1. Here the **CEO** is 1 as it is at the top, he has no manager/boss at it's top and we access all the other employees using the CEO only.
2. The Employee **4** has two pointers subordinate_one and subordinate_two, they point to employees with id **6** and id **7** respectively.
3. The employee with id **5** has its subordinate_one as **NULL** but it's subordinate_two points to a employee with id 9.
4. All the ids are non-negative and unique.
5. The Interns i.e. - `6,7,8,9` have both their subordinates as **NULL**

If the input for the company was `5 2 -1 -1 3 -1 -1` then the company would be the follows. The first value is always the CEO, followed by the subordinate_one team. The subordinate_one team starts at 2 and has both pointers as **NULL** as the input is -1 and the team with subordinate_two on top starts at 3 with both pointers as **NULL**. Here the actual input to the starter code is `5 Gate 10000 2 Nadela 5000 -1 -1 3 Ballmer 6000 -1 -1`. Here there will be three employees. The employee with **id** as 5 will have his name as **Gates** and his salary as **10000**. Similarly

Another example is `5 2 6 -1 -1 -1 3 -1 1 -1 -1` then the hierarchy would be -



The first value is the CEO then subordinate_one's team input will start at **2**, then we will give the CEO's subordinate_one's subordinate_one's id with value **6**. **6** has both pointers **NULL** as the two next numbers are **-1**. Similarly the CEO's subordinate_two team is made. You **do not** have to handle creation of the company as that is done by the starter code. You only have to use this structure for implementation of the functions. The above explanation is so that yo can give your own custom input in the `input.txt` file

Functions to be implemented are as below:

`FUNCTIONS`:

1. `double get_average_salary(int emp_id)`:
   This function takes a parameter *emp_id* as **Employee ID** and return the average salary of the team which he is the team head.(Remember: CEO is the head of the company).

2. `void get_all_bosses(int emp_id)`:
   This function takes the parameter *emp_id* as **Employee ID** and prints names of all the team heads of the teams in which employee is working starting from the CEO till his boss. If there is no boss, print -1. **For ex. -** if we call `get_all_bosses(9)` in the Fig1. then it would print names of the employees with IDs 1,2,5 **all in same line with spaces)**.

3. `void same_last_names(int emp_id)`:
   This function takes the parameter *emp_id* and prints the employee ids of all the employees in the company, who have the same last name as of the employee with id *emp_id* (including him) **all in same line with spaces**. You must print them according to there **ascending order of their levels.** If two person have the same level and same last name then you can print them in any order.

4. `int get_first_common_boss(int emp_id1,int emp_id2):`
   This function takes two parameters `emp_id1` and `emp_id2` and returns the first common boss of both the employees. The first common boss is defined as the team head of the team, with least number of employees of which both employees are part of.

Important Notes:

1. You can import Lists from previous assignment (Assignment 10), if needed. We are releasing the solutions, you can make the necessary changes and include it in the headers file or maybe copy-paste the code. If you include in the header file you must include the other file in the submission as well or it would not run.
2. **The emp_id will always be an id of employee in the company and we will not give random values here.**
3. The level may not be valid.

Below we provide a description of the `input.txt` file that is given to you along with starter-code for more clarity. You may change the file to test your code on various other inputs. We have also provided a `Makefile` that enables you to compile your code and then run it on the given `input.txt` file. Just use the command `make` on the terminal. For more information on makefiles, refer here and here.

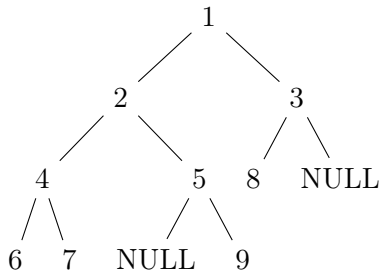
`INPUT`:

```
1 smith 10000 2 kumar 2000 4 gates 400 6 bansal 5000 -1 -1 7 jain 67 -1 -1 5 yadav 22 -1 9 kumar
    789 -1 -1 3 taylor 67 8 smith 111 -1 -1 -1
2 7
3 get_first_common_boss 1 4
4 get_first_common_boss 6 9
5 same_last_names 1
6 get_average_salary 2
7 get_all_bosses 6
8 get_all_bosses 1
9 get_all_bosses 2
```

`OUTPUT`:

```
1
2
1 8
1379.67
1 2 4
-1
1
```

EXPLANATION The structure that gets built is -



1. the first common team in which both 1 and 4 work is the organization itself so team head of the organization is 1 so print.
2. the first common team which 6 and 9 work is the team with head 2 so first common boss is 2.
3. last names of employee 1 is 'smith' and employee 8 is also 'smith' and as level of employee 1 is less so we print 1 8**(each in a new line)**.
4. the team of which 2 is the head has employees $2, 4, 6, 7, 5, 9$. Hence the average salary calculated turns out to be 1379.67.
5. team head of the teams which 6 works under is 4 and then if we go up a level then team head of the team in which 6 work in is 2 and if we again go up a level then we reach the organization and team head of the organization is CEO - 1. So we print in the ascending order of the levels so finally we print 1 2 4.
6. Since no level above CEO so we print -1.
7. team head of the team which 2 works under is 1.