COL100 Assignment 11
2^{nd} Semester Semester : 2021-2022

Deadline: 11:59 pm, 19 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

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. Each employee (including me) may be in charge of at most two people. An employee of my organisation has an **EmployeeID (unique in the organisation and 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.

1. **Employee:** This is a structure in C that contains **EmployeeID** 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.

In this assignment you will be implementing this organisational hierarchical structure in C programming language and some functions on them. You are given this structure 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:

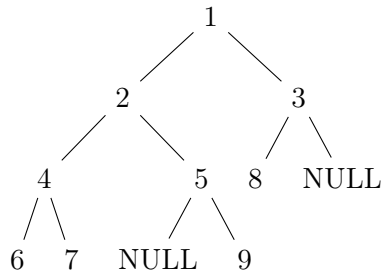
1. `create_employee(int x)`: This function would create a new `Employee` with the `id` attribute as `x` and the `subordinate one` and `subordinate two` attribute as `NULL` and return a pointer to it. Note that the creation of the structure is done by us and **we will provide only non negative unique 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.

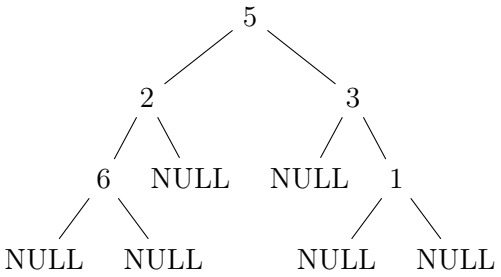


Consider the above company, the input file will contain the list of employee IDs `1 2 4 6 -1 -1 7 -1 -1 5 -1 9 -1 -1 3 8 -1 -1 -1` , based on this list the `create_company()` will create the given company hierarchical structure. Here we have followed the convention that at first we have the employee id 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. This order will always be unique for a company and we handle the creation of 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**.

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. `void get_employees()`: In-Lab Component
   In this function you have to traverse the company i.e. you have to print all the employees in the company in a particular order explained below.

   Algorithm to traverse a team -

   (a) Print the team_head.
   (b) Traverse the subordinate_one team
   (c) Traverse the subordinate_two team

   For Ex- `get_employees()` when called on the last company given above will return `5 2 6 3 1`. You need not add a newline character at the end as that is done by the driver code. You do not have to return anything. Note that in this all of the employee ids will be printed therefore do not skip any employee. **Note: You do not have to print -1 for NULL pointers**

2. `int Distance(int emp_id1, int emp_id2)`
   This function takes two ids $id1$ and $id2$ and will return the distance between the ids. A distance is the count of total number of edges between employee with EmployeeID id1 to employee with EmployeeID id2. The distance is always unique for 2 employees and will be zero if the id's are same. You do not have to print anything here and only the value has to be returned.

   Distance is the absolute count of the edges between the Employees with the given ids. **For ex -** The distance between **5** and **6** in the last company is 2 and the distance between **6** and **1** is 4. It is **NOT** the difference in level.

3. `void ImmediateTeam(int emp_id)`:
   This function prints the immediate team of an employee - it includes their boss and the subordinate_one and subordinate_two i.e. the left and right pointers contained in that employee. The printing format is [boss] [subordinate_one] [subordinate_two], with one space between employees. You do not need to add the newline character as that is done by the starter code.

   Note: You do not have to print -1 for NULL pointers

4. `int Level(int emp_id)`:
   This function returns the level of an employee with the given id. The level of an employee is the number of edges present in path from the CEO to that employee. You do not have to print anything, just return the value of the level. Note that CEO has level 0.

5. `void EmployeesAtSameLevel(int level)`:
   This function prints all the employee ids at a given level, note that root is at level 0.

   The printing format is left to right i.e. subordinate_one and then subordinate_two with one space separation between employees. The output should be single line and you need not add the newline character. In any of the functions which involve printing you need not print -1 for NULL pointers.

   For Ex: Consider the last company hierarchical structure defined above. The employees at level 0 are `5`, the employees at level 1 are `2 3` and the employees at level 2 are `6 9`

   Note: You do not have to print -1 for NULL pointers

6. `int Boss(int emp_id)`: In-Lab Component
   This function takes an employee id this will belong to an employee in the company and your task is to return the id of its boss/manager. You do not have to print anything.
   Note: If the boss does not exists return -1

7. `int Diameter()`:
   This function returns the diameter of a company - the diameter is the maximum distance between any two ids in the company. You can use the Distance function implemented above. **Note:** You do not have to print anything.

8. `int TeamSize(int emp_id)`:
   This function takes an id of an employee and returns the number of employees the person works with i.e. the size of his immediate team. It is defined as the number of employees connected to it via edges. NULL pointers are not included. You do not have to print anything.

Important Note: 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.

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.

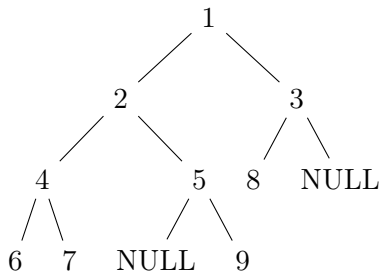
`INPUT`:

```
1 2 4 6 -1 -1 7 -1 -1 5 -1 9 -1 -1 3 8 -1 -1 -1
15
level 1
level 5
distance 4 5
distance 6 8
distance 1 1
employees_at_same_level 2
get_emmployees
immediate_team 4
immediate_team 5
immediate_team 8
team_size 9
team_size 4
diameter
boss 6
boss 1
```

`OUTPUT`:

```
0
2
2
5
0
4 5 8
1 2 4 6 7 5 9 3 8
2 6 7
2 9
3
1
3
5
4
-1
```

`EXPLANATION` The structure that gets built is -



1. The level of employee with id **1** is 0 as it is the CEO
2. The level of employee with id **5** is 2 as there are two edges that we have to move from, or you could say there are two hops that we have to make from the CEO **1** to the Employee **5**.
3. The distance between **4** and **5** is 2 as we have to make two hops - one from **4** to **2** and then from **2** to **5**. Note in this company you cannot visit a boss/manager of an employee therefore you need to remember who the boss was.

4. The distance between **6** and **8** is 5 as there are 5 edges in between.

5. The distance between **1** and **1** is 0.

6. The numbering of levels starts from 0, therefore the employees at level 2 are `4,5,8` with the ordering from the left most to the right most.

7. get_employees will do a search of all employees. We visit the CEO and print it. Then we visit the subordinate_one team. It will keep looking up subordinate_one teams i.e. visit subordinate_one until it reaches a null pointer and then it looks up the subordinate_two. So it keeps visiting subordinate_one until it reaches employee with id **6** till this it prints the values `1 2 4 6` then it visits the subordinate_two value of **4** to print **7**. Similarly then it visits the subordinate_two team of **2** and then the subordinate_two team of **1**. **Hint:** the helper function `print_company()` does something similar.

8. The immediate_team of employee 4 are `2,6,7` in the order - **boss, subordinate_one, subordinate_two**

9. The immediate_team of employee 5 are `2,9`

10. The immediate_team of employee 8 are `3`

11. The team_size of 9 is **1** as it is connected to only one employee.

12. The team_size of 4 is **3** as it is connected to 3 employees.

13. The diameter of the company is the maximum distance between any two employees therefore the value is 5. It can be said to be the distance between **6 and 8** or between **7 and 8**, etc

14. The boss of 6 is 4

15. The boss of 1 is -1 as there is no boss