Implement a C application for managing tasks that were assigned to software engineers. For this you will use a HashTable that will have as useful information an element of type pointer to a Task (Task\*); structure Task has the following attributes: *task id* (**unsigned** **int**), *task date* (**char\***), *owner’s name* (**char\***), and *task completion* (**float – percentage level**).

1. Create a HashTable with at least 7 entries/tasks. The data for each entry is read from a text file. The search key used by the HashTable is the attribute ***owner’s name***. ***(1 p)***
2. Print out to the console all the elements in the HashTable. ***(0.5 p)***
3. Write the function that determines the number of tasks for a given owner specified as a parameter. The function is called in the **main()** and the result is displayed on the console. ***(1 p)***
4. Write the function for counting the number of tasks with the completion level above a certain value specified as a parameter. The function is called in the **main()** and the result is displayed on the console. ***(1 p)***
5. Write the function that changes the owner of a given task, specified by task id. The function is called in the **main()** and the result is validated at the console by displaying all the elements. ***(1 p)***
6. Write the function that returns a list with all the tasks that were assigned on a given date, specified as a parameter. The list doesn’t share HEAP memory space with the elements found in the HashTable. The function is called in the **main()** and the result (elements stored in the list) is displayed on the console. ***(1.5 p)***

***The following indications are mandatory:***

***Functions that are not tested in the main() function are not considered for evaluation; All implementations must be called in the main() function;***

***Projects with compilation issues are NOT going to be evaluated;***

***Source code that is commented is NOT going to be evaluated;***

***The implementation must NOT trigger memory leaks;***