CS 201, Fall 2023
Homework Assignment 1

Due: 23:59, October 29, 2023

In this homework, you will implement a simple issue-tracking system for an imaginary company. Every task that needs to be completed in the company is named an "issue". As a company policy, they assign each issue to only one employee. The company may hire new employees, as well as employees may leave the company. In your implementation, you **MUST** use dynamically allocated arrays for storing the employee and issue information.

The issue tracking system will have the following functionalities. The details of these functionalities are given below:

1. Add a new employee (i.e., a new employee is hired)
2. Remove an employee (i.e., an employee left the company)
3. Create an issue and assign it to an employee
4. Remove an issue
5. Change assignee of all issues of an employee
6. Show the list of all employees
7. Show the list of all issues
8. Show detailed information about an employee
9. Show detailed information about an issue

Add a new employee: The issue-tracking system will allow the company to add a new employee indicating their name and title. The employee names must be unique in the system. Thus, the system should check whether or not the specified employee already exists, and if they exist, it should not allow the operation and display a warning message. Initially, an employee is not assigned any issues.

Remove an employee: The issue-tracking system will allow the company to remove an existing employee indicated by their name. If the employee does not exist, the system should not allow the operation and should display a warning message. If the employee has some assigned issues, the system should not allow the operation and should display a warning message.

Create an issue: The issue-tracking system will be able to create an issue by giving ID number of the issue, description of the issue, and name of the assignee. Issue IDs are unique, there should be no more than one issue with the same ID in the system, at the same time. If an issue with the same ID already exists in the system, the system should not allow the operation and display a warning. Note that an ID can be reused after its issue is removed.

Remove an issue: The issue-tracking system will be able to remove an issue by giving ID number of the issue. If the issue does not exist, the system should not allow the operation and should display a warning message. After the issue is removed, its ID number is now available and could be used for a new issue.

Change assignee: The issue-tracking system will allow to move issues from one employee to another. In this case, all of the issues of the previous assignee will be assigned to the new assignee. As a result, the number of issues assigned to the previous employee will become zero. If the previous and/or new assignee names do not exist, the system should not allow the operation and display a warning message.

Show the list of all employees: The issue-tracking system will allow to display a list of all employees. The list includes, for each employee, the employee's name, title, and the number of issues assigned. Note that the order of employees in the list IS important; you should list in the order they join the company.

Show the list of all issues: The issue-tracking system will allow to display of a list of all issues. The list includes, for each issue, the issue's ID, description, and name of the assignee. Note that the order of issues in the list IS important; you should follow the employee order (the order they join the company), and among issues of the same employee, you should follow the assigning order. Note that, in change assignee operation, the assigning order is preserved (e.g., if the previous assignee has 2 issues assigned and new assignee has 5 issues before the change assignee operation, the first issue of the previous employee will be 6th issue of the new assignee and second issue of previous assignee will be 7th issue of the new assignee.)

Show detailed information about an employee: The issue-tracking system will allow to specify the name of an employee and display detailed information about that particular employee. This information includes the employee's name, title, and number of issues assigned to that employee. If the employee with that name does not exist, the system should display a warning message.

Show detailed information about an issue: The issue-tracking system will allow to specify the ID of an issue and display detailed information about that particular issue. This information includes the issue's ID, description, and name of the assignee. If an issue with that ID does not exist, the system should display a warning message.

Below is the required `public` part of the `IssueTrackingSystem` class that you must write in this assignment. The name of the class must be `IssueTrackingSystem`, and must include these public member functions. The interface for the class must be written in the file called `IssueTrackingSystem.h` and its implementation must be written in the file called `IssueTrackingSystem.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution and implement them in separate files.

```
class IssueTrackingSystem {

public:
    IssueTrackingSystem();
    ~IssueTrackingSystem();

    void addEmployee( const string name, const string title );
    void removeEmployee( const string name );
    void addIssue( const int issueId, const string description, const string
        assigneeName );
    void removeIssue( const int issueId );
    void changeAssignee( const string previousAssignee, const string newAssignee );

    void showAllEmployees() const;
    void showAllIssues() const;
    void showEmployee( const string name ) const;
    void showIssue( const int issueId ) const;
};
```

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `IssueTrackingSystem`, its interface is in the file called `IssueTrackingSystem.h`, and the required functions are defined as shown above. Your implementation should use the format given in the example output to display the messages expected as the result of the defined functions.

Example test code:

```
#include <iostream>
using namespace std;

#include "IssueTrackingSystem.h"

int main() {

    IssueTrackingSystem ITS;

    ITS.showAllEmployees();
    cout << endl;

    ITS.addEmployee( "Alper", "Software Engineer" );
    ITS.addEmployee( "Enes", "Software Engineer" );
    ITS.addEmployee( "Ayse", "Software Engineer" );
    ITS.addEmployee( "Salih", "Software Engineer" );
    ITS.addEmployee( "Salih", "ML Engineer" );
    ITS.addEmployee( "Fatma", "Software Engineer" );
    ITS.addEmployee( "Yusuf", "Cook" );
    ITS.removeEmployee( "Kamil");
    ITS.removeEmployee( "Enes");
    ITS.addEmployee( "Merve", "Researcher" );
    ITS.addEmployee( "Ali", "DevOps Engineer" );
    ITS.removeEmployee( "Ali" );
    cout << endl;

    ITS.addIssue( 1, "Code payment module", "Ayse" );
    ITS.addIssue( 2, "Code login page", "Salih" );
    ITS.addIssue( 3, "Code logout page", "Fatma" );
    ITS.addIssue( 4, "Code search engine", "Ayse" );
    ITS.addIssue( 5, "Create algorithm for new recommendation engine", "Merve" );
    ITS.addIssue( 6, "Create robots.txt to inform crawlers", "Alper" );
    ITS.addIssue( 7, "Code profile pages", "Alper" );
    ITS.addIssue( 1, "Fix image not found bug", "Merve" );
    ITS.addIssue( 2, "Fix login bug", "Salih" );
    ITS.removeIssue( 8 );
    ITS.removeIssue( 1 );
    cout << endl;

    ITS.changeAssignee( "Hasan", "Huseyin" );
    ITS.changeAssignee( "Hasan", "Salih" );
    ITS.changeAssignee( "Salih", "Fatma" );
    ITS.removeEmployee( "Salih" );
    ITS.removeEmployee( "Fatma" );
    ITS.removeIssue( 4 );
    ITS.removeEmployee( "Ayse" );
    cout << endl;

    ITS.showAllEmployees();
    cout << endl;
    ITS.showEmployee( "Fatma" );
    cout << endl;
    ITS.showEmployee( "John" );
    cout << endl;
```

```
    ITS.showAllIssues();
    cout << endl;
    ITS.showIssue( 1 );
    cout << endl;
    ITS.showIssue( 3 );

    return 0;
}
```

Output of the example test code:

```
Employees in the system:
None

Added employee Alper.
Added employee Enes.
Added employee Ayse.
Added employee Salih.
Cannot add employee. Employee with name Salih already exists.
Added employee Fatma.
Added employee Yusuf.
Cannot remove employee. There is no employee with name Kamil.
Removed employee Enes.
Added employee Merve.
Added employee Ali.
Removed employee Ali.

Added issue 1.
Added issue 2.
Added issue 3.
Added issue 4.
Added issue 5.
Added issue 6.
Added issue 7.
Cannot add issue. Issue with ID 1 already exists.
Cannot add issue. Issue with ID 2 already exists.
Cannot remove issue. There is no issue with ID 8.
Removed issue 1.

Cannot change assignee. Previous or/and new assignee does not exist.
Cannot change assignee. Previous or/and new assignee does not exist.
Salih's issues are transferred to Fatma.
Removed employee Salih.
Cannot remove employee. Fatma has assigned issues.
Removed issue 4.
Removed employee Ayse.

Employees in the system:
Alper, Software Engineer, 2 issue(s).
Fatma, Software Engineer, 2 issue(s).
Yusuf, Cook, 0 issue(s).
Merve, Researcher, 1 issue(s).

Fatma, Software Engineer, 2 issue(s).

Cannot show employee. There is no employee with name John.
```

```
Issues in the system:
6, Create robots.txt to inform crawlers, Alper.
7, Code profile pages, Alper.
3, Code logout page, Fatma.
2, Code login page, Fatma.
5, Create algorithm for new recommendation engine, Merve.

Cannot show issue. There is no issue with ID 1.

3, Code logout page, Fatma.
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use dynamically allocated arrays with only the necessary amount of memory in your implementation. That is, if there are 10 employees in the system, it should use memory only for these 10 employees. In other words, you cannot initially allocate a large array for employees and expect it to get filled later. The same argument applies to space used to store issues. You will get no points if you use fixed-sized arrays, linked lists or any other data structures such as vectors/arrays from the standard library. However, if necessary, you may define additional data members and member functions.
2. Moreover, you ARE NOT ALLOWED to use any global variables or any global functions.
3. Output message for each operation MUST match the format shown in the output of the example code.
4. Your code MUST NOT have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct. To detect memory leaks, you may want to use Valgrind which is available at `http://valgrind.org`.

NOTES ABOUT SUBMISSION:

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. Your class name MUST BE `IssueTrackingSystem` and your file names MUST BE `IssueTrackingSystem.h` and `IssueTrackingSystem.cpp`. Note that you may write additional class(es) in your solution.
2. The code (`main` function) given above is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you MUST NOT submit these test codes (we will use our own test code). In other words, do not submit a file that contains a function called `main`.
3. You should put all of your `.h` and `.cpp` files into a folder and zip the folder (in this zip file, there should not be any file containing a `main` function). The name of this zip file should conform to the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number. The submissions that do not obey these rules will not be graded.
4. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

5. You are free to write your programs in any environment (you may use Linux, Windows, MacOS, etc.). On the other hand, we will test your programs on "dijkstra.ug.bcc.bilkent.edu.tr" and we will expect your programs to compile and run on the dijkstra machine. If we could not get your program properly work on the dijkstra machine, you would lose a considerable amount of points. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

6. This assignment is due by 23:59 on Sunday, October 29, 2023. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. Please see the course home page for further discussion of academic integrity and the honor code for programming courses in our department.

8. For questions regarding academic integrity and use of external tools, please refer to the Honor Code for Introductory Programming Courses (CS 101/102/201/202) at `https://docs.google.com/document/d/1v_3ltpV_lClLsROXrMbojyuv4KrFQAm1uoz3SdC-7es/edit?usp=sharing`.

9. **This homework will be graded by your TA Burak Ferit Aktan (ferit.aktan at bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.**