CS 201, Fall 2024
Homework Assignment 1

Due: 23:59, October 30, 2024

In this homework, you will implement a space mission management system. The system will manage various space missions and spacecrafts. Each mission and spacecraft must have a unique name. Each mission can have multiple assigned spacecrafts but each spacecraft can only be assigned to one mission at a time. The management system should support adding and removing missions and spacecraft, as well as assigning spacecrafts to missions. In your implementation, you **MUST** use dynamically allocated arrays for storing the mission and spacecraft information.
The space mission management system will have the following functionalities. The details of these functionalities are given below:

1. Add a mission to the system.
2. Remove a mission from the system.
3. Add a spacecraft to the system.
4. Remove a spacecraft from the system.
5. Assign a spacecraft to a mission.
6. Drop a spacecraft from a mission.
7. Show the list of all missions.
8. Show the list of all spacecrafts.
9. Show detailed information about a specific mission.
10. Show detailed information about a specific spacecraft.

Add a mission to the system: The management system will allow the user to add a mission, indicating its name, launch date, and destination. The mission names must be unique in the system. Thus, the system should check whether or not the specified mission already exists, and if it exists, it should not allow the operation and display a warning message. Initially, a mission does not have any assigned spacecraft.

Remove a mission from the system: The management system will allow the user to remove an existing mission indicated by its name. If the mission does not exist, the system should not allow the operation and display a warning message. If the mission has spacecrafts assigned, all assigned spacecrafts' status should be updated as available so they could be assigned to other missions when needed.

Add a spacecraft to the system: The management system will allow the user to add a new spacecraft, specifying its name and type. The system should ensure that the spacecraft name is unique. If a spacecraft with the same name already exists, it should prevent the addition and display a warning message. Initially, the spacecraft is "Available".

Remove a spacecraft from the system: The management system will allow the user to remove an existing spacecraft. If the spacecraft does not exist, the system should not allow the operation and display a warning message. If the spacecraft is already assigned to a mission (i.e., its status is assigned), the system should not allow the operation and display a warning message.

Assign a spacecraft to a mission: The management system will allow the user to assign a spacecraft to a mission. Once the spacecraft is assigned, its availability status is updated as "Assigned". If the spacecraft does not exist, the system should not allow the operation and display a warning message. If the spacecraft exists but the mission does not exist, the system should not allow the operation and display a warning message. If the spacecraft exists but is not available, the system should not allow the operation and display a warning message.

Drop a spacecraft from a mission: The management system will allow the user to specify the name of a spacecraft to drop it form its mission if it has any. If the spacecraft is not assigned to any mission, the system should display a warning message. If the spacecraft is assigned to a mission, it should remove the spacecraft from that mission and update the spacecraft's status to available.

Show the list of all missions: The management system will display a list of all missions. The list should include the mission name, launch date, destination, and the number of assigned spacecrafts. Note that the order of missions in the list IS important; you MUST list in the order they are added to the system.

Show the list of all spacecrafts: The management system will display a list of all spacecrafts. The list should include the spacecraft name, type, and status. Note that the order of spacecrafts in the list IS important; you MUST list in the order they are added to the system.

Show detailed information about a specific mission: The management system will allow the user to specify the name of a mission and display detailed information about that mission, including the name, launch date, destination, and assigned spacecrafts if any. If the mission does not exist, the system should display a warning message. Note that the order of spacecrafts in the list IS important; you MUST list in the order they are assigned to the mission.

Show detailed information about a specific spacecraft: The management system will allow the user to specify the name of a spacecraft and display detailed information about that spacecraft, including its name, type, and status. If the spacecraft does not exist, the system should display a warning message.

Below is the required `public` part of the `SpaceMissionManagementSystem` class that you must write in this assignment. The name of the class must be `SpaceMissionManagementSystem`, and must include these public member functions.
The interface for the class must be written in the file called `SpaceMissionManagementSystem.h` and its implementation must be written in the file called `SpaceMissionManagementSystem.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 SpaceMissionManagementSystem {

public:
    SpaceMissionManagementSystem();
    ~SpaceMissionManagementSystem();

    void addMission( const string name, const string launchDate, const string
        destination );
    void removeMission( const string name );
    void addSpacecraft( const string name, const string type );
    void removeSpacecraft( const string name );
    void assignSpacecraftToMission( const string spacecraftName, const string
        missionName );
    void dropSpacecraftFromMission( const string spacecraftName );
```

```
    void showAllMissions() const;
    void showAllSpacecrafts() const;
    void showMission( const string name ) const;
    void showSpacecraft( const string name ) const;
};
```

Here is an example test program that uses this class and the corresponding output. We will use similar programs to test your solution so make sure that the name of the class is `SpaceMissionManagementSystem`, its interface is in the file called `SpaceMissionManagementSystem.h`, and the required functions are defined as shown above. Your implementation MUST use EXACTLY the same 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 "SpaceMissionManagementSystem.h"

int main() {

    SpaceMissionManagementSystem SMS;

    SMS.showAllMissions();
    cout << endl;

    SMS.showAllSpacecrafts();
    cout << endl;

    SMS.addMission("Apollo 11", "1969-07-16", "Moon");
    SMS.addMission("Mars Exploration", "2020-07-30", "Mars");
    SMS.addMission("Jupiter Orbiter", "2030-01-01", "Jupiter");
    SMS.addMission("Apollo 11", "1969-07-16", "Moon");
    cout << endl;

    SMS.showAllMissions();
    cout << endl;

    SMS.removeMission("Voyager 5");
    SMS.removeMission("Jupiter Orbiter");
    cout << endl;

    SMS.showAllMissions();
    cout << endl;

    SMS.addSpacecraft("Orion", "Crewed");
    SMS.addSpacecraft("Voyager 1", "Uncrewed");
    SMS.addSpacecraft("Curiosity", "Rover");
    SMS.addSpacecraft("Enterprise", "Crewed");
    SMS.addSpacecraft("Voyager 1", "Uncrewed");
    SMS.addSpacecraft("Voyager 2", "Rover");
    cout << endl;

    SMS.showAllSpacecrafts();
    cout << endl;
```

```
    SMS.removeSpacecraft("Hubble");
    SMS.removeSpacecraft("Orion");
    cout << endl;

    SMS.showAllSpacecrafts();
    cout << endl;

    SMS.assignSpacecraftToMission("Voyager 1", "Mars Exploration");
    SMS.assignSpacecraftToMission("Voyager 2", "Mars Exploration");
    SMS.assignSpacecraftToMission("Curiosity", "Apollo 11");
    cout << endl;

    SMS.showAllMissions();
    cout << endl;

    SMS.showAllSpacecrafts();
    cout << endl;

    SMS.removeSpacecraft("Voyager 1");
    SMS.removeSpacecraft("Curiosity");
    cout << endl;

    SMS.assignSpacecraftToMission("Orion", "Apollo 11");
    SMS.assignSpacecraftToMission("Enterprise", "Saturn Voyage");
    SMS.assignSpacecraftToMission("Orion", "Saturn Voyage");
    SMS.assignSpacecraftToMission("Voyager 1", "Mars Exploration");
    SMS.assignSpacecraftToMission("Curiosity", "Mars Exploration");
    SMS.assignSpacecraftToMission("Enterprise", "Mars Exploration");
    cout << endl;

    SMS.showMission("Mars Exploration");
    cout << endl;

    SMS.dropSpacecraftFromMission("Voyager 1");
    SMS.dropSpacecraftFromMission("Orion");
    SMS.dropSpacecraftFromMission("Curiosity");
    SMS.dropSpacecraftFromMission("Voyager 1");
    cout << endl;

    SMS.showAllMissions();
    cout << endl;

    SMS.showAllSpacecrafts();
    cout << endl;

    SMS.showMission("Apollo 11");
    cout << endl;

    SMS.showMission("Voyager 10");
    cout << endl;

    SMS.showSpacecraft("Voyager 2");
    cout << endl;

    SMS.showSpacecraft("Hubble");
    cout << endl;
```

```
    SMS.removeMission("Mars Exploration");
    cout << endl;

    SMS.showAllSpacecrafts();
    return 0;
}
```

Output of the example test code:

```
Missions in the space mission management system:
None

Spacecrafts in the space mission management system:
None

Added mission Apollo 11.
Added mission Mars Exploration.
Added mission Jupiter Orbiter.
Cannot add mission. Mission Apollo 11 already exists.

Missions in the space mission management system:
Mission: Apollo 11, Launch Date: 1969-07-16, Destination: Moon, Assigned Spacecraft
    Count: 0
Mission: Mars Exploration, Launch Date: 2020-07-30, Destination: Mars, Assigned
    Spacecraft Count: 0
Mission: Jupiter Orbiter, Launch Date: 2030-01-01, Destination: Jupiter, Assigned
    Spacecraft Count: 0

Cannot remove mission. Mission Voyager 5 does not exist.
Removed mission Jupiter Orbiter.

Missions in the space mission management system:
Mission: Apollo 11, Launch Date: 1969-07-16, Destination: Moon, Assigned Spacecraft
    Count: 0
Mission: Mars Exploration, Launch Date: 2020-07-30, Destination: Mars, Assigned
    Spacecraft Count: 0

Added spacecraft Orion.
Added spacecraft Voyager 1.
Added spacecraft Curiosity.
Added spacecraft Enterprise.
Cannot add spacecraft. Spacecraft Voyager 1 already exists.
Added spacecraft Voyager 2.

Spacecrafts in the space mission management system:
Spacecraft: Orion, Type: Crewed, Status: Available
Spacecraft: Voyager 1, Type: Uncrewed, Status: Available
Spacecraft: Curiosity, Type: Rover, Status: Available
Spacecraft: Enterprise, Type: Crewed, Status: Available
Spacecraft: Voyager 2, Type: Rover, Status: Available

Cannot remove spacecraft. Spacecraft Hubble does not exist.
Removed spacecraft Orion.

Spacecrafts in the space mission management system:
```

```
Spacecraft: Voyager 1, Type: Uncrewed, Status: Available
Spacecraft: Curiosity, Type: Rover, Status: Available
Spacecraft: Enterprise, Type: Crewed, Status: Available
Spacecraft: Voyager 2, Type: Rover, Status: Available

Assigned spacecraft Voyager 1 to mission Mars Exploration.
Assigned spacecraft Voyager 2 to mission Mars Exploration.
Assigned spacecraft Curiosity to mission Apollo 11.

Missions in the space mission management system:
Mission: Apollo 11, Launch Date: 1969-07-16, Destination: Moon, Assigned Spacecraft
    Count: 1
Mission: Mars Exploration, Launch Date: 2020-07-30, Destination: Mars, Assigned
    Spacecraft Count: 2

Spacecrafts in the space mission management system:
Spacecraft: Voyager 1, Type: Uncrewed, Status: Assigned
Spacecraft: Curiosity, Type: Rover, Status: Assigned
Spacecraft: Enterprise, Type: Crewed, Status: Available
Spacecraft: Voyager 2, Type: Rover, Status: Assigned

Cannot remove spacecraft. Spacecraft Voyager 1 is assigned to a mission.
Cannot remove spacecraft. Spacecraft Curiosity is assigned to a mission.

Cannot assign spacecraft. Spacecraft Orion does not exist.
Cannot assign spacecraft. Mission Saturn Voyage does not exist.
Cannot assign spacecraft. Spacecraft Orion does not exist.
Cannot assign spacecraft. Spacecraft Voyager 1 is already assigned to mission Mars
    Exploration.
Cannot assign spacecraft. Spacecraft Curiosity is already assigned to mission Apollo
    11.
Assigned spacecraft Enterprise to mission Mars Exploration.

Mission:
  Name: Mars Exploration
  Launch Date: 2020-07-30
  Destination: Mars
  Assigned Spacecrafts:
  - Voyager 1
  - Voyager 2
  - Enterprise

Dropped spacecraft Voyager 1 from mission Mars Exploration.
Cannot drop spacecraft. Spacecraft Orion does not exist.
Dropped spacecraft Curiosity from mission Apollo 11.
Cannot drop spacecraft. Spacecraft Voyager 1 is not assigned to any mission.

Missions in the space mission management system:
Mission: Apollo 11, Launch Date: 1969-07-16, Destination: Moon, Assigned Spacecraft
    Count: 0
Mission: Mars Exploration, Launch Date: 2020-07-30, Destination: Mars, Assigned
    Spacecraft Count: 2

Spacecrafts in the space mission management system:
Spacecraft: Voyager 1, Type: Uncrewed, Status: Available
Spacecraft: Curiosity, Type: Rover, Status: Available
```

```
Spacecraft: Enterprise, Type: Crewed, Status: Assigned
Spacecraft: Voyager 2, Type: Rover, Status: Assigned

Mission:
  Name: Apollo 11
  Launch Date: 1969-07-16
  Destination: Moon
  Assigned Spacecrafts:
    None

Cannot show mission. Mission Voyager 10 does not exist.

Spacecraft: Voyager 2, Type: Rover, Status: Assigned

Cannot show spacecraft. Spacecraft Hubble does not exist.

Removed mission Mars Exploration.

Spacecrafts in the space mission management system:
Spacecraft: Voyager 1, Type: Uncrewed, Status: Available
Spacecraft: Curiosity, Type: Rover, Status: Available
Spacecraft: Enterprise, Type: Crewed, Status: Available
Spacecraft: Voyager 2, Type: Rover, Status: Available
```

IMPORTANT NOTES:
Do not start working on 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 missions in the system, it should use memory only for these 10 missions. In other words, you cannot initially allocate a large array for missions and expect it to get filled later. The same argument applies to space used to store spacecrafts. **You will get no points if you use fixed-sized arrays, linked lists or any other data structures such as vectors/arrays/etc. 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. Your output will be compared verbatim with the expected output during evaluation.
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`.
5. You may assume that the inputs for the functions are in correct format (e.g., launch date of a mission is a valid date) so that you do not need to make any input checks for format.
6. Both mission and spacecraft names should be unique and case-sensitive.

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 `SpaceMissionManagementSystem` and your file names MUST BE `SpaceMissionManagementSystem.h` and `SpaceMissionManagementSystem.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. Your code will be tested by using an automated test suite that includes multiple test cases where each case corresponds to a specific number of points in the overall grade. We will provide you with example test cases by email. Thus, we strongly recommend you to make sure that your program successfully compiles and correctly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment. If your current code does not fully compile on dijkstra before submission, you can try to comment out the faulty parts so that the remaining code can be compiled and tested during evaluation.

6. This assignment is due by 23:59 on Wednesday, October 30, 2024. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. Late submissions will not be accepted (if you can upload to Moodle, then you are fine). There will be no extension to this deadline.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. For questions regarding academic integrity and use of external tools (including generative AI tools), please refer to the course home page and 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`.

8. **This homework will be graded by your TA Sude Önder (sude.onder at bilkent.edu.tr). Thus, you may ask your homework related questions directly to her. There will also be a forum on Moodle for questions.**