ENSC 251 Project – Phase 1: Interplanetary Alliance Recruitment System

Mission Overview

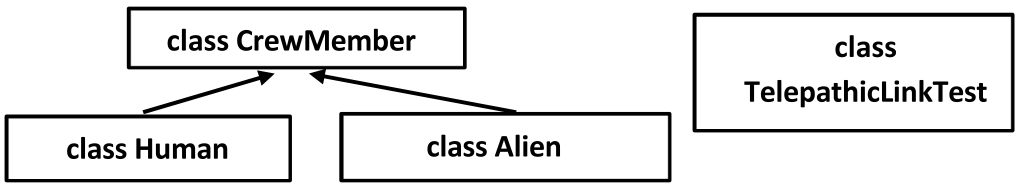
In this first phase of the ENSC 251 project, you will design and implement a simplified recruitment evaluation system for the Interplanetary Alliance. The objective is to apply Object-Oriented Programming (OOP) in C++ to model, process, and rank potential recruits. This handout preserves the original ENSC 251 structure and marks while rewriting the prose for clarity.

The total project weight is 50 marks: 20 marks for Phase 1 (this interim report) and 30 marks for Phase 2 (final). Teams of four students will be assessed on design quality, correctness, testing depth, and coding discipline. Detailed rubrics will not be released before marking.

Divide the work roughly based on the listed tasks. The overall marks for major tasks are listed for your guidance, but not the detailed marks for every single item. Please include a simple document in the project report indicating which parts each member finished. There is some negative marking if you don't complete the listed tasks; details are not listed.

Object Modeling and Class Structure

Model two categories of recruits—Humans and Aliens—using four C++ classes. Both share common fields but include origin-specific data.



• Base: CrewMember — first name, last name, TrainingScore (float, 0–4.3), MissionAptitude (int, 0–100), and an 8-digit ID (start 25220000).

• Derived: Human — adds home sector (string).

• Derived: Alien — adds homeworld (string) and a TelepathicLinkTest object.

• Auxiliary: TelepathicLinkTest — subscores: signalStrength, cognitiveLink, empathicOveraly, cognitiveResistance (each 0–10) and total (sum).

Phase 1 Tasks and Evaluation (Numbered Requirements; 20 Marks Total)

Use numbered steps as a guide for implementation and verification. Only the items with a [marks] tag contribute directly to the 20-mark total; other numbered items are required specifications that affect your grade.

1. [4 marks] Class Design and ADTs — Implement CrewMember, HumanCrew, AlienCrew, and TelepathicLinkTest as abstract data types with appropriate constructors, getters/setters, and invariants. Place declarations in headers and definitions in .cpp files (e.g., crewmember.hpp / crewmember.cpp). Ensure correct inheritance and encapsulation.

2. [2 marks] Friend Comparison Functions — In CrewMember, implement: compareTrainingScore, compareMissionAptitude, compareFirstName, compareLastName. Each takes (const CrewMember& a, const CrewMember& b) and returns less-than / equal / greater-than.

3. [1 mark] Stream Output — Overload operator << for Human and Alien to display clear, labeled fields for a single recruit.

4. [1 mark] Data Input and Storage — Read records from human-crew.txt and alien-crew.txt. After parsing each line, construct the appropriate object and store HumanCrew and AlienCrew in separate vectors.

5. [4 marks] Single-Field Sorting — On user request, sort either Human or Alien by exactly one field: TrainingScore, MissionAptitude, FirstName, or LastName.

- Numeric fields sort high→low; names sort A→Z.
- Sorting functions accept an unsorted vector and return a sorted copy (do not mutate inputs).
- Implement sorting code in crew_sort.hpp / crew_sort.cpp. Library sort APIs are not allowed (see Requirement 10). Algorithmic efficiency counts.

6. [4 marks] Overall Multi-Field Sorting — Implement a total ordering applied within a single category at a time (humans only or aliens only):

a) Primary key: MissionAptitude (descending).
b) Secondary key: TrainingScore (descending).
c) Tertiary key: Home sector (humans) or Homeworld (aliens), alphabetical.
d) For AlienCrew, exclude recruits with TelepathicLinkTest total < 15.

7. Interactive Menu — Your program presents a looped menu to select: (i) single-field sort (human or alien), (ii) overall sort (human or alien), and other tasks as applicable. Continue until the user chooses to exit.

8. Basic Error Checking — Validate input formats and ranges during parsing; handle malformed lines gracefully (skip with a message) without crashing the program.

9. [3 marks] Testing and Validation — Demonstrate coverage of: normal cases, boundary/extreme values (e.g., ties, max/min scores), and invalid inputs. Show evidence of expected ordering and correct filtering for TelepathicLinkTest thresholds.

10. **No Library Sort** — Do not use std::sort or container-specific sort APIs. Implement your own efficient algorithms. Vector and string APIs are permitted.

11. Coding Standards — Use consistent naming, meaningful comments, modular structure, and readable formatting. Non-idiomatic or confusing code will be penalized.

12. Feel free to add additional modules to make your code more modular and easier to manage.

13. [1 mark] One-Page PDF — Include a concise PDF describing:

a) team task distribution
b) test plan summary
c) weekly schedule and progress tracking

14. Please make sure your code can compile and run correctly on the lab computer. If your code cannot compile, then your group can get at most 10 marks. If your code can compile, but cannot run, then your group can get at most 12 marks.

Supplied data files

You will be given two files: human-crew.txt and alien-crew.txt. The first line is a heading for each field. For the rest of the lines, each line represents the data for one crew with comma separated data for field. The human-crew.txt and alien-crew.txt will be different in the number of fields as they contain different data.

There will be some deliberate errors in the files. Ignore these for Phase 1.

Submission

All files to be submitted to via Canvas and demonstrated to TAs. Deadlines TBD.