Project Guidelines

You must follow these guidelines for all the projects that you work on in the data structures course.

1. **Submissions**
   You need to submit the following three items:
   - **Source code:** Compress your project folder. The folder must consist of all your source code files. I will use this folder to run your application. Submit the source code on blackboard.
   - **Project Report**: This report contains assumptions, a UML diagram, Big-O of the significant functions, and references.
   - **Individual Report**: A short report (preferably one page only) for each team member. Each team member should mention his/her contributions, and what was learned in the project, other comments. Further, each person should briefly mention the other team member's contributions. **You might lose 25% of the grade if you don't submit this report.**
     Here's the template for the individual report if you are working in a group:
     https://www.dropbox.com/s/duki6ga6x2cnw6v/personal_report.docx?dl=0
     If you are working solo, here's the template for the individual report:
     https://www.dropbox.com/s/5sstebqneubw6aa/personal_report_solo.docx?dl=0

2. **Submission Guidelines**
   - Submit everything on Blackboard (no need for hard copies).
   - **Teams:** Each team member should submit his/her individual report separately on Blackboard. The team leader should submit the source code and the project report on Blackboard (one submission per team). **Please don't e-mail me the submissions. Make the submissions on Blackboard.**
   - **Solo:** Submit all the required submissions on Blackboard.

3. **Source Code**
   - You can use C++, Java, or C# to implement your project.
   - You need to submit all the files that are needed to run the project. In C++, for instance, you will need to submit header files (.h) and cpp files.
   - Comment your code, and choose meaningful variable and function names.
   - Follow a coding style consistently.
   - Don't include the source code in your printed report. I will just look at it electronically.

4. **Report**
   This report should be roughly between 3-4 pages (excluding the front-page).

   Here is what your report should contain:

   - **Assumptions:** This section is optional. List any assumptions that you made that aren't mentioned of the project description.
   - **A UML class diagram:** The diagram will show the classes you used, and the relationships between them.

- **Efficiency of Algorithms:** Mention the Big-O of the required functions (just the significant functions, not minor things like setters and getters). Mention if it is possible to do better.
- **References:** Cite any references that you used.

5. **Examples for Inspiration**
   - The phone directory project: You can find it on Blackboard in *Course Content->Source Code->Vectors.*
   - The airline system: You can find it in the textbook, page 381. However, a sequence diagram (page 383) is **not** required.

6. **Teamwork**
   - Each team should appoint a leader. It is your responsibility as a team to delegate responsibilities to team members. Make sure the responsibilities are relatively equal. Each team member must have a role in coding.
   - **At any point in time, if there is an issue with your team, contact me as early as possible**. I can't help you when you inform me about problems too late (a day or two before the deadline).
   - The idea behind teamwork is that you exchange thoughts with each other. Hence, teach other about your contributions.
   - Each team member's performance will be evaluated based on the contributions that are mentioned in the individual report.
   - **A team member who doesn't contribute or contributes poorly will most likely get a very poor grade.**

7. Grading Policy (for teams)

Factor	Percentage
Whether the source code can compile, run, and do what is required.	10%
Class diagram	10%
Fulfillment of the technical requirements	20%
Quality of algorithms: • Good performance of the functions. • Well-documented source code. • Meaningful variable names. • Object-oriented source code.	30%
Individual contribution to the team (Each team member gets a different mark)	25%
Teamwork	5%

8. Grading Policy (Solo)

Factor	Percentage
Whether the source code can compile, run, and do what is required.	10%
Class diagram	10%
Fulfillment of the technical requirements	40%
Quality of algorithms: • Good performance of the functions. • Well-documented source code. • Meaningful variable names. • Object-oriented source code.	40%