CSE 4202 - Structured Programming II Lab - Project Plan

Brief	Structured Programming Lab 2 final project should be a simple team project developed in C or C++
Team Configuration	3 Members per team, should be from the same lab group
Language: C and C++	Before going in depth, there are huge differences between C and C++ projects, since a lot of necessary data structures can be easily used in C++ via STL, which needs to be hand crafted in C. A better solution is to use industry standard libraries like GLIB, but such cannot be expected from a 1st year student. So the project will definitely be evaluated based on the implementation language.
Project Idea and Acceptance	First of all, it is a 3 person project. So very simple projects that can be whipped up in a day or two will not be accepted. Definitely, the project can be simple and should be simple, but it needs to be elaborate. That means, different members need to provide different functionalities in the single project, which is discussed in the examples. The project idea submission is a 2 phase process. Firstly, before the end of the final exam, all the teams must be formed and come up with ideas to showcase. Tentatively, at the very first lab after the mid term examination, the project ideas of each of the teams will be checked. The students **must give a presentation (will be marked)** to express their ideas and the different features they are going to implement. Alongside this, they must give an overview, how they are going to divide their work for the project. It can change, but an initial rough idea will help the teams to divide their work. If the ideas are proper, they will be accepted. Otherwise, modifications will be suggested accordingly. If required, students can (and should) use **slides**, documents, and/or drawings to express their ideas.
Project Examples	**1) Management Applications:** This type of application mainly serves management of information. For instance, a command-line based diary application can be taken into account. The diary application can serve many purposes. For instance, the user can write notes for a specific day of the year. The user can make a schedule and get it displayed. There can also be a simple dashboard when the application starts to show upcoming tasks. The user might be able to filter prior notes based on date/time criteria or search for different notes based on keywords. Management application's internals can be divided into three major parts, a file or database system that takes care of serialization/deserialization. A set of functions that implements different features like creation of entry, efficient searching and finally a front end, that views different information and facilitates user interaction. Since it is a three person project, each team should come up with enough features to justify their effort. These last few statements hold true for all the different kinds of projects.
	2) Utility Applications (Tool): A utility application or tool is generally designed to assist its user in certain specific tasks. A prime example would be the simple application that I used to set up the questions for the very first quiz of the structured programming course in the previous semester. Basically that application has a corpus or pool of individual question scripts. Each question script has the question, its answer and a couple of tags attached to it. For simplicity, tags can be like "easy", "medium" and/or "hard". In the frontend, the user can give a set of rules, like for every individual student, give 2 "easy" questions, 2 "medium" questions and 1 "hard" question. Ofcourse, for a single individual, the same question must not be repeated. After the criteria, the user also gives the system a list of student ID and student names. For my implementation, I supplied it in CSV format. It is a simple interpretable format. The question scripts were in MarkDown (GFM) format. Once these things are mentioned, the system creates questions and solutions for each individual based on the rules and student list.
	3) Simple GUI Applications: As already mentioned before, making a GUI application in C is pretty hard. A GUI application can contain all the features mentioned above, but with a Graphical User Interface. Now, if it is about industry standards, for C, the framework to use would be GTK+ and for C++, it would be QT. But they are for higher level usage. If someone knows C++ very well, he/she with his/her team can move to QT. However, there are many simple alternatives. If using C, students can use the IGraphics library to implement simple GUI applications and games. A list of projects can be found at https://github.com/topics/igraphics-project. Tutorials are available online to teach the students how to deal with IGraphics. For C++, a better alternative would be to use SFML. Now SFML is pretty powerful and the list is not provided here, because students can just pick a simple application of their liking type and find examples on the internet.
	4) Simple Games: Games are always kind of an attraction for the first year students and they can actually be a great source of fun and learning. Now the games can be divided into two sections again. Some are purely command line based and can get pretty complex. For instance, a popular card game can easily be developed, where the player plays their hand on each turn. Now, while designing such games where there will be opponents, the students also need to think whether they are going to implement some simple rule based AI or make in turn based multiplayer. In this category, table top or turn based RPG, quiz games with twist (just a word for more features), choice based stories can also be included. The next category would be commandline games, but with clever graphics, For instance, a game of Tetris, Pacman or Snake can be implemented in the command-line while using Block Characters and continuous screen clearing to show each frame individually. ~~You have already seen such an implementation for Conway's game of life.~~ You just need to figure out how to decouple inputs and keep the game updating at a constant rate. These are operating system specific. It might require some looking up, but is not hard to do. One important factor is that codes for such games are widely available online. Students are encouraged to look online to learn and find solutions, but blindly implementing them without understanding will never be accepted. In short, students do need to explain what they have done. In the final section, there are graphical games. As stated already, it is pretty difficult to implement graphics related frameworks in C, for the first year students. Still if they want, they can look into simple frameworks like SFML and implement simple graphical games.
Project Progress Update	After the acceptance of the idea, sometime between the second half of the semester, a project progress update may take place, where each individual team needs to report their progress so far and their plan ahead. A presentation might take place.
Final Project Evaluation	The students need to PRESENT their works at the end of the semester labs. Each team and individual will be evaluated based on some criteria. Each member's contribution, their understanding of the modules they have implemented, the different features and very importantly, how well a team has managed its code, how well the code is written and how easily it is understandable. Documentation will also be appreciated a lot in the case of complex codes. Students are encouraged to give a simple presentation to demonstrate what their projects are before showcasing it. It gives them a stage to show what motivated them to do their project and how they have tackled the issues they faced while developing the project.
Mark Division	**Total Mark :** 45 (30%) **Breakdown:** **Implementation:** 15 **Demonstration:** 15 **Viva/Explanation:** 15