Operating Systems Problem Proposal (Deadline: Sep 2, 2013, 11.59 PM)

Project Title:	Enter the project title.	
Prepared By:	Enter the name of the person completing the proposal.	**Date:**

Project Team Size and Members:
Size (4-5) and Names (along with rolls and email-id):

Problem Background (in context of specific OS design issue):

Problem Statement: Briefly summarize the problem you propose to address. This statement provides basis for rest of the document. Write this in terms of OS problem being addressed, not in terms of the solution needed, or implementation proposed.

Purpose Statement (Goals):

This is what you intend to do to address the problem identified above. This is the solution you have decided to implement. A project has one goal that gives purpose and direction to the project. This will be used as a continual point of reference for any questions that arise regarding scope or purpose. This section should be written in language that is easy for everyone to understand. It describes what will be implemented, corrected, installed, replaced or otherwise addressed to solve the problem.

Objectives/Deliverables: (To be submitted in the document due for submission by Sep 21, 2013 11.59 pm)

- Objectives and deliverables are a more detailed version of the purpose (goals) statement. They outline what will be accomplished in this project.
- Objective statements will clarify boundaries of the purpose statement and define boundaries of the scope of the project.
- Every objective must be accomplished in order to reach the goal and accomplish the purpose of this project.
- Consider including as objectives / deliverables:
  - Your requirements gathering process
  - Assumptions and Constraints (including any specific use of file formats)
  - A formal implementation plan
  - Testing strategy (including a discussion on input and output)
  - Hand-offs / recommendations to other teams

Methods/Approach:

- How will you complete this project? What is your strategy for completion?
- What tools will you use?
- Will this project change or impact the understanding of any particular OS concept? How?

Success Criteria:

- This is the measurable value resulting from doing this project.
- What state must exist for your teacher to say that the project was a success?

Test Plan and Integration:

- Interfaces to other systems/implementations/projects, and
- Online/real-time input, including data from presently used manual forms/running OS etc.

Risks and Dependencies along with Assumptions/Deficiencies:

- Identify any factors that can affect the outcome of the project including major dependencies on other events/teams.
- These factors can affect deliverables, success, and completion of the project.
- Record anything that can go wrong during this project and the probability.
- Discuss deficiencies, including limitations such as time delays.
- Describe any assumptions and constraints that will affect development and operation of your implementation. Identify any limitations affecting the desired capability, any desired capabilities that will not be provided by the proposed implementation, as well as any anticipated operational changes that will affect the proposed operation of the system.

Operating Systems Problem Proposal (Deadline: Sep 2, 2013, 11.59 PM)

Required Consultations: The following consultations may be useful for any project. Identify which consultation should be required for your project:

- Functional requirements from the team leads, and any related issues from the architects.
- Technical requirements with the course teacher, and/or the architects.

Resources:

- People – team lead and other participants.
- Time - what is the time frame for this project? How long will it take?
- Other – software, expertise, etc.

Project Duration (in weeks – you can complete in a time lesser than assigned to your team):

What goes into a design document?

A design document is a complete high-level solution to the problem presented. It should be detailed enough that somebody who already understands the problem could go out and code the project without having to make any significant decisions. Further, if this somebody happens to be an experienced coder, they should be able to use the design document to code the solution in a few hours.

So what actually goes in addition to the description to the above mentioned?

- A high-level description of your solution, including **design decisions** and **data structures**
- **Declarations** for all new classes, procedures, and global/class variables
- **Descriptions** of all new procedures (unless you can tell exactly what it does from the name), including the purpose of the procedure, and an explanation of how it works and/or pseudocode

Also, another important thing to remember is that a design document needs to include the **correctness invariants** and **testing strategy**. The testing strategy includes a clear plan of the testing methodology and may include a description of test cases that will be used to test correctness invariants. Focus on the testing strategy. If you want, you may itemize things that will need testing.

What *doesn't* go into a design document?

Do not restate the problem in your design document. We are far more interested in your solution than in knowing that you understood the problem.

Your design document should contain very little actual code, if any at all. Include psuedocode for all complex procedures, but do not include Java/C/C++ code.

The purpose of pseudocode is to avoid all the annoying aspects of programming languages that make code both harder to write and harder to read. The purpose of pseudocode is *not* to be imprecise about how you solve a problem. Comments are welcome.

Remember that we have to *read* your design documents. If you don't think we want to see it, don't put it in!

Comments on the Design

What this design document gets right:

- **Good use of pseudocode.** Design document submissions SHOULD NOT include verbatim Java code; you will lose points for this. Rather, provide clear and concise pseudocode describing the primary algorithms and any special data structures used in your implementation.
- **Use of text descriptions.** It usually suffices to describe the primary algorithms and data structures of your solution in a short paragraph, then provide more specifics in the pseudocode.
- **Length.** In general, the length of a design document section should increase in proportion to the difficulty and length of the solution it describes.

What this design document does poorly? (based on past experiences)

- **TEST CASES!** We need to see that you've actually considered how you're going to test your implementation.

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	5/30/00	Design Document Template and Checklist
Rev. 1	6/6/00	Additions to Section 7
Rev. 2	4/10/02	Conversion to WORD 2000 format