

The final project is intended to give you some experience applying the materials covered in this course to loosely defined medium sized problem in a small team setting. In general, all projects must satisfy the following basic criteria:

- A brief project proposal.
- A presentation to the class.
- Code development.
- Some sort of software demonstration.
- A written report explaining your use & selection of algorithms & container classes.
- Team work.
- Additional components as needed (charts, etc)

The four main components (and their associated goals) of the project are:

- Project proposal: clarify goals and expectations for your project.
- Midpoint check-in: solve problems before they become last minute nightmares.
- Project presentation: demonstrate your project to the class.
- Final report: document and summarize your project & turn in all deliverables.

Included at the end of this document is a list of potential project ideas. You are free to come up with your own ideas, but you'll need to run them past me before you proceed with them. In general, I'll want to ensure that your project can demonstrate course knowledge and has an appropriate level of difficulty.

Things to keep in mind:

The project is an alternative to a final comprehensive 3 hour exam – you should gauge your work accordingly.

Attempting to field a decent scoring project at the last minute will be somewhere between improbable and impossible. I've tried to design the project to discourage procrastination, but you'll need to make an earnest effort to ensure success.

Every project needs to be different in some way (also extends to previous semesters) – check with me if you have a topic that overlaps another group's work.

Project ideas:

Predator / prey simulation.

Design a predator / prey simulation. For example, you might have hawks & mice. We can assume the mice can 'live off the land,' while hawks can only eat mice. Both move around & reproduce randomly based on parameters.

Disk scheduling comparisons.

Research several disk scheduling algorithms & build a simulation. Run a series of trials & track their performance stats.

Process scheduling.

Research process scheduling algorithms & build a simulation. Run a series of trials & track their performance stats.

Abstract Game.

Research an abstract game. Some recommendations are: Ataxx, Reversi, Nine Men's Morris, Connect6, and Connect4. Implement some manner of AI (computer opponent) functionality. The AI doesn't need to be perfect, but it should do more than make random moves.

Travel Planner.

Create a travel planner as described in text on page 746. Add functionality to force include or exclude intermediate cities; I.E. shortest path from LA to NY that passes through Moorhead, but not Denver.

Maze / Dungeon generator.

Create a program to generate mazes & dungeons. You could add parameters to influence the maze (lots of turns, few 3 ways intersections, etc) and / or modify the maze into a dungeon which contain rooms. You may find it helpful to research 'roguelikes'.

Inventory packer.

Write a program to spatial pack items into an inventory. For example, we might want to pack two 1x3 & three 2x2 rectangles into a 4x4 area. Note: this problem gets significantly trickier if you allow the items to be rotated.

Crossword Generator.

Create a program that takes a list of strings as input & generates a crossword type puzzle out of them. Write another program to take a list of the words & a puzzle and find a solution.

Map Searching.

Create a program to generate random maps with multiple features (mountains, water, forest, etc). Develop search routines to find areas that satisfy multiple constraints; I.E. find a 3x4 section with water, trees & no mountains.