# Lesson 4.XX — Programming Project (MagPie Alternative)

## Overview

This project is intended to be an alternative to the MagPie Lab in the original AP computer science course. Starting from 2014-2015 the College Board announced that 20 hours of hands-on lab time may replace the original required labs (MagPie, PictureLab, Elevens). In this document, we recommend some open-ended alternatives.

---

### Objectives — *Students will be able to…*

- **Conduct user-centered research** to identify specific functions for a specialized application
- **Plan and create** a calculator that perform specialized operations for an end-user
- **Test, evaluate, and share** the end product

### Assessments — *Students will…*

- **Apply** if-else, String methods to implement a software application
- **Submit** a complete, functional program

### Pacing Suggestion - The duration of this project is at the discretion of the teacher. We recommend 2 hours for class time design and implementation, and 1-2 hours for out of class time to connect with end-user. - Project involves conducting research work (survey or interviews), and communicating with end-user, out of the classroom.

## Implementation Details

### Complexity and Creativity

Students should come up with the idea themselves, based on user-centered research, and ideate a specialized application to address the needs of a specific user group.

Students will follow applied design process to implement the idea. You should talk to your teacher often to ensure that your progress is in-line with expectations. ### Documentation and Style As with all projects, your program must be well-written, well-documented, and readable. Writing code with good style is always good idea. This will help you debug, pick up where you left off each day, and keep track of progress.

STEP 1 - UNDERSTANDING CONTEXT

Conduct user-centered research to find design opportunities and barriers.

Select an end-user for whom you will design and create this program (this can be a friend, classmate, relative, etc). Create interview questions that will allow you to understand the end-user's interests and likes/dislikes.

Here are some possible applications that could potentially use "if" statements and "String" methods:

**Word Games** * Lipogram Word Game https://en.wikipedia.org/wiki/Lipogram * A Mad-Libs Program https://en.wikipedia.org/wiki/Mad_Libs * A acrostic builder pyramid word builder http://www.crauswords.com/pyramidword.h

**Classic Ciphers with a Twist** (eg, Morse Code, Caesar Cipher,etc.) https://en.wikipedia.org/wiki/Classical_cipher * Create your own encryption program, but include some decision making (eg. Ask user for clues/hints for key generation?) * Create the corresponding decryption program, but include some decision making * Work in pairs! Test with one another.

**Chatbot** https://en.wikipedia.org/wiki/Chatbot * One of the four Activities in the original MagPie Lab (estimated 1-2 hour time) * A specialized, modern ChatBox (Siri/Alexa) for a specific end-user in mind: - TV show related

- music DJ - learning tool for ELL, French immersion, or other foreign language students - someone who is color blind - someone who owns a pet store

**Other** * Other application that involves STRINGS (creation, parsing, and processing), together with decision making (IF statements) - Medical terminology taxonomy - language learning applications - creating baby books or random poetry

After you have some initial ideas, interview some potential end-user to clarify your project specifications.

---

STEP 2 - DEFINING AND IDEATING

Choose a design opportunity and point of view, make inferences about limitations and boundaries. Take creative risks to identify gaps to explore, generate a range of possibilities, prioritize ideas for prototyping.

Using the responses from your end-user interview, begin to develop a plan for your custom application. Given the duration of the project, you should limit your calculator to do *an interesting activity* for a *specific end-user*. At this point you will only be be to implement a text-based user interface (i.e., no graphical elements, like buttons or scrollbars). Keep it simple and easy to use.

List: * the types of calculator functions it will do: * what will the user input be? * what output will be? * what user interaction with the program will the user have? * how do you start the activity? * how does the activity end?

Share these ideas with your end user. Record their comments, suggestions and feedback and note any changes that you may make as a result of this interview.

Identify any issues or problems that might arise as you begin to program (what areas might require more information or programming solutions? Where can you find this information?)

STEP 3 – PROTOTYPING AND TESTING

Construct prototypes, making changes to code as needed.

Program your calculator. Be sure to review course notes and activities to make sure that you effectively implement object oriented programming design for your application.

When you are ready, create a short test plan and test your program. Include expected output and actual output and include details related to any fixes that needed to be made. A good motto to remember is **code a little, test a little**

Have your end-user try a working version of your program. Note their suggestions, comments and feedback. Indicate any changes that you made to the program, based on the user's feedback.

STEP 4 – SHARING, TESTING AND FINAL ITERATION

Gather feedback from users over time to critically evaluate your design and make changes to product design or processes Identify new design issues.

Share your work with other classmates, friends or family. Record any feedback, suggestions and comments and use this information to make the final iteration of your program.

| Grading Scheme/Rubric | |
|---|---|
| Implementation | |
| Project is appropriately complex and creative | 4 points |
| Program is well-documented and shows good style | 10 points |
| Final product meets all requirements and goals laid out in checkpoint specifications | 8 points |
| Program uses programming concepts effectively, including all required elements with an appropriate level of complexity | 4 points |
| Object Oriented Programming concepts are effectively applied with an appropriate level of complexity | 6 points |