# Software Project I: Algorithms

## Purpose

To create (a) product(s) that engage(s) you and that you would be proud to share to a public audience.

Along the way, you will develop your ability to problem-solve using a variety of strategies, to design an algorithm, to implement a solution in code, to manage source code using accepted industry practices, and to plan and meet commitments for project milestones.

## Evaluation

75% of your final grade on a product comes from your ability to provide regular evidence that you have met curriculum expectations in this course.

Using your final commit GitHub, and your posts on Sesame, what have you learned and demonstrated knowledge of?

You probably will not have demonstrated all of the expectations listed, but have you hit the majority of the expectations?

How well? Did you develop the ability to meet these expectations independently? Did you challenge yourself?

## Curriculum Expectations I Believe I Have Met

### A1. Data Types and Expressions Demonstrate the ability to use different data types, including one-dimensional arrays, in computer programs;

**A1.1** use constants and variables, including integers, floating points, strings, and Boolean values, correctly in computer programs;

ASCII, Unicode) to internally represent data and store information;

Eg. <https://github.com/rsgc-noblecurveira-c/algorithms-isp/blob/master/encoder-isp.playground/Contents.swift#L12-L44>

**A1.3** use assignment statements correctly with both arithmetic and string expressions in computer programs;

**A1.4** demonstrate the ability to use Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and order of operations correctly in computer programs;

**A1.5** describe the structure of one-dimensional arrays and related concepts, including elements, indexes, and bounds;

**A1.6** write programs that declare, initialize, modify, and access one-dimensional arrays.

### A2. Data Types and Expressions Demonstrate the ability to use control structures and simple algorithms in computer programs;

**A2.1** write programs that incorporate user input, processing, and screen output;

**A2.2** use sequence, selection, and repetition control structures to create programming solutions;

**A2.3** write algorithms with nested structures (e.g., to count elements in an array, calculate a total, find highest or lowest value, or perform a linear search).

### A3. Subprograms Demonstrate the ability to use subprograms within computer programs;

**A3.1** demonstrate the ability to use existing sub-programs (e.g., random number generator, substring, absolute value) within computer programs;

**A3.2** write subprograms (e.g., functions, procedures) that use parameter passing and appropriate variable scope (e.g., local, global), to perform tasks within programs.

### A4. Code Maintenance Use proper code maintenance techniques and conventions when creating computer programs.

**A4.1** demonstrate the ability to identify and correct syntax, logic, and run-time errors in computer programs;

**A4.2** use workplace and professional conventions (e.g., naming, indenting, commenting) correctly to write programs and internal documentation;

**A4.3** demonstrate the ability to interpret error messages displayed by programming tools (e.g., compiler, debugging tool), at different times during the software development process (e.g., writing, compilation, testing);

**A4.4** use a tracing technique to understand program flow and to identify and correct logic and run-time errors in computer programs;

**A4.5** demonstrate the ability to validate a program using a full range of test cases.

### B1. Problem-solving Strategies Use a variety of problem-solving strategies to solve different types of problems independently and as part of a team;

**B1.1** use various problem-solving strategies (e.g., stepwise refinement, divide and conquer, working backwards, examples, extreme cases, tables and charts, trial and error) when solving different types of problems;

**B1.2** demonstrate the ability to solve problems independently and as part of a team;

**B1.3** use the input-process-output model to solve problems.

### B3. Designing Algorithms Use a variety of problem-solving strategies to solve different types of problems independently and as part of a team;

**B3.1** design simple algorithms according to specifications.

## Final Comments and Proposal for Level of Achievement

Taking into consideration the purpose of this project and the evaluation criteria, what overall level of achievement do you feel you have earned?