# 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;

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L24-L52>

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

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L364>

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L91-L108>

**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;

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L117-L368>

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

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L202-L211>

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

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L202-L211>

### 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;

Yes, the entire program does this.

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

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L238-L368>

**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).

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L238-L368>

### 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;

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L364>

**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.

The entire program shows this.

### 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;

As seen through my github commits, I have correctly written the syntax, and this involved overcoming many errors. In order to do this, when I would first come across an error, I would try to logically get around it by re-arranging variables, as well as changing data types to the correct type. If this didn’t work, I would search up the error online to discover if it was common, and a method for overcoming it.

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

The entire code shows this.

**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);

In order to get over many struggles, I had to determine what the error message was telling me. One such case involved my random colour generator, which at first didn’t work. The error message told me that a UInt32 couldn’t be turned into an Int in the manner which I had tried, so I rearranged the line such that it worked.

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

I displayed multiple things using the print function, in order to show me what variables were doing at particular times. This helped me overcome many errors, and is still in the program if it “Hack Colours” option is selected

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

My entire program has 7 different examples (the different functions) and 1 colour change example of this.

### 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;

Yes, this was done all throughout the creation of this code in order to come to the final completed code.

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

All the people at our table helped each other to overcome all problems as they arose.

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

Seen in my IPO entrance.

### 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.

<https://github.com/rsgc-Burleton-J/Algorithms---ISP/blob/master/Animation/Sketch.swift#L222>

## 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?

I feel that I did very well given the circumstances. I was given the task to turn my basic function generator and turn it into the basis of an app (2 screens). Keeping this in mind, I was able to fix all the glitches with my program and add a user interface in two days. For this project, I not only created multiple algorithms, but also used graphics in order to incorporate user input into the program.

I hope I was able to surpass expectations with this project, as I found it thoroughly challenging, and was incredibly proud when I was able to complete it.