High level goals

1) Perseverance implies success; self-actualization (motivation, self aware, self propelling, goal setting)

Learning outcomes

- Look up how to do something you didn’t know how to do
- Work with a partner to solve a problem
- When faced with a problem to solve, articulate the scope and details of the problem
- Identify one step to make progress towards a solution (here and with the last one I’m trying to get at the fact that students shouldn’t just throw up their hands when they don’t know what to do. They should take steps to make progress toward a solution. We’ll want to refine this and phrase it better)
- Develop goals for open-ended (programming) projects
- Go beyond the basic requirements of an assignment
- Employ one or more effective strategies to cope with frustration
- Articulate the knowledge and skills acquired over the 4-week program
- Overcome an obstacle to accomplish something you are proud of
- Approach computational/mathematical problems with confidence
- Describe yourself as someone who is capable of mastering material through hard work (embrace the growth mindset)
- Accurately identify your strengths and areas for improvement in the context of the SPIS program, in enough detail that you can work toward improvement (related: articulate ways to improve)
- Estimate how much time a task will take (this is probably not something we’ll be able to accomplish, but is there something related we can include?)
- Consider long-term goals and research and pursue useful stepping stones to achieve them
- Be aware of what resources might help you succeed and take advantage of them
- Connect current activities and coursework to a wider context (of the discipline) and, specifically, to the role they play in your life goals

2) Critical thinking, algorithmic problem solving, implementing and engineering a solution

Learning outcomes

Programming

- Use proper Python syntax to create a program that runs
- Describe the different data types in Python, and how to convert between data types
- Use conditional statements in Python programs
- Use recursion and loops in Python programs
- Write programs to manipulate images
- Use Python’s bitwise operators to secretly embed and recover information within an image

-

<u>Algorithm development (perhaps this should just join programming)</u>
- Identify base case and recursive step in a recursive algorithm
- Write algorithms to solve recursive problems, articulating the base case and the recursive step

<u>Debugging/reasoning about code</u>
- Articulate the expected behavior of a program (including variable values, and execution flow)
- Insert print statements to trace execution flow of a program
- Insert print statements to display values of variables
- Trace Python programs to describe their behavior (including values of variables)
- Develop test cases to verify the correctness of Python programs
- Draw box-diagrams (and box-and-arrow diagrams) to trace the values of variables and other data through assignment statements and other data mutation
- Draw stack frames to model the execution of functions in Python

<u>Proofs</u>
- Identify a proof by induction and distinguish between regular and strong induction
- Prove a statement (equality, inequality) by induction, making appropriate choices for the base case(s) and induction step(s)

<u>Definitions</u>
- Define basic math concepts: integers, natural numbers, rational numbers, real numbers, divisibility, prime numbers, function, set, sequence

<u>Mathematical Manipulations</u> (I don't really like this wording--sorry)
- Perform basic integer operations, including those involving modular arithmetic
- Convert a number between binary, decimal, hex
- Understand and construct sums using Sigma notation
- Convert negative integers to two's complement binary representation
- Write a program to convert numbers between binary, decimal, and hex
- Develop geometric formulae to relate Pixels to each other to produce a desired image effect
- Develop bit-level equations (shifts, and, or, addition, etc) to secretly embed information within and image

- Work with recursive definitions of sets, sequences, functions, constructions
- Define new sets, sequences, or functions using recursive definitions

- Identify strings matched by regular expressions
- Perform arithmetic operations on numbers represented in binary or hex
- Describe the connection between the base of the representation and logarithm and multiplication
- Describe the challenges behind representing real numbers in computers

3) Analytical thinking and writing, framing useful larger questions about technology, humanity, and culture

What we plan to do to teach this:
- In feedback sessions, give examples and discuss gold standard previous work as well as lower quality work and ask the students what differences they see between them.

4) Smooth transition to UCSD CS (I think we need our tutors' help with this one!)
- Articulate curricular and academic expectations for CSE majors
- Describe who to go to for various issues (which issues?)

4a) Understanding of the scope of CS, and excitement about the discipline
- Articulate several real-world problems CS researchers and practitioners might study and solve
- Express excitement (relevance?) about CS (really not sure about this one).

What we plan to do to facilitate this
- CS Tutors will act as mentors to small groups of SPIS students
- Assignments will use UCSD lower division standards (ACMS usernames, turnin scripts, drafting)
- In Facets: Presentations by UCSD CSE advisors
- In Facets: Laura Stevens about Calculus sequence, ??? about Humanities.

* Communicate expectations and goals to students
* SPIS graduates buddying up with students in the Fall
* SPIS students giving feedback on program
* How to teach what are the standards?

ASSESSING SUCCESS OF SPIS:
- Pre and Post Student Survey
-- Include questions that try to "measure" the extent to which students articulate what they do and do not know?
---- e.g. What are CS / Math concepts you'd like to know more about?
---- e.g. What are CS / Math concepts you understand well?

- In Post Student survey
-- Include student self evaluation of progress
-- Instructors will comment on this on Completion Certificate with individualized summary feedback.
- Control group: this year, compare performance of SPIS students in their first quarter (first year) courses with that of students who were invited to SPIS but didn't attend (similar population pool).
- For future years, can build more careful control groups to isolate some aspect of SPIS to be evaluated (individual attention, programming experience, programming language used, etc.).

Pre-test and Post-test drafts:

1. <Concrete Math / CS question, perhaps multipart and perhaps beyond the scope? Maybe just background question>

Example: relatively prime numbers are useful in various contexts. Two natural numbers are relatively prime if their only common positive factor (divisor) is 1. How would you test if two given numbers are relatively prime?

Alternatively: A robot is programmed to travel around a grid whose cells are labelled by their x and y coordinates (x,y). The robot starts at (0,0) and at each step is allowed to move only diagonally. For example, after the first step, the robot may be at (1,1) or at (1,-1) or at (-1,1) or at (-1,-1). Can the robot ever reach the coordinates (1,0)? Explain your reasoning.

2. What is Computer Science (CS)? ** This doesn’t seem to tie in to any of the high-level goals above. We might think about adding it, since I think that one of the goals is to give students an idea and an excitement about CS. I added something, but we should talk about it.
3. Describe one thing someone who studies or works in CS might study or do?
4. How important do you think CS or programming skills will be in your future? (This one might be interesting, since I”m not sure this is an explicit goal.... yet).
5. Rank your familiarity with technology

resident debugger ---> use them for schoolwork and socializing but haven’t looked under the hood → try to avoid!

6. Rank your familiarity / comfort level with computer science
7. Rank your familiarity with “debugging”
8. Rank your familiarity / comfort level with mathematics
    - (7) I have a good foundation in both concepts and computation and am able to make new conjectures and test them
    - (4) I am good at computations but am sometimes unsure about what the result of the computation means

    - (1)

9. Rank your familiarity / comfort level with writing
10.

11. Rank your study skills (motivation, time management, goal setting)

12. Rank your familiarity with UCSD

Post-test only:
1-7 (for each aspect of SPIS) stimulated my interest in the subject matter
1-7 I learned a great deal during SPIS
What personal SPIS accomplishments are you most proud of?
What do you feel was challenging for you during SPIS?
What study skills were most useful for you during SPIS?

<u>Pre test notes:</u>

Knowledge/understanding of CS
Personal interest in CS
Experience with CS
Experience with Math
Confidence solving technical problems
Confidence programming
Confidence in Math
Writing experience
Writing confidence
Are you a native English speaker
Confidence in success transitioning to UCSD
Study skills/time management
Growth vs. fixed mindset of learning
Approach to solving a technical problem (the robot problem--ask also what steps they took to solve it)