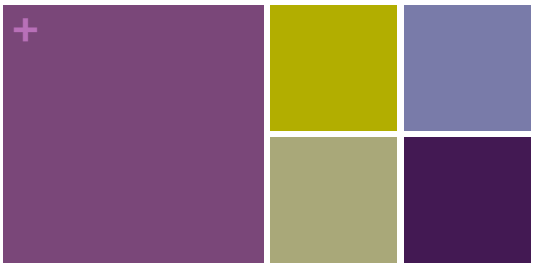

Testing and
Program Design
Ex: Encoded Messages

The Stateless Function Paradigm

- A stateless function is a function that does not store state information between calls to the function.
- A stateless function only acts on its input, constants, and values/variables derived from the computation that the function does.
- Given the same input, a stateless function should always act the exact same way.

Why should we write stateless functions?

- Self contained
- Deterministic outcome
- Easy to test

Testing: Brainstorm Individual Tests

- What are the things that the function was written to do?
- What are typical ways that the function would be called?
- What preconditions could be violated by the caller?
- How could the function be misused?
- What values are you expecting as input?
- What values are you not expecting as input?
- What are the edge cases or exceptional conditions?

Testing: Writing your test

- Create sample data and expected results
  - Do not look at the code before writing your expected results.
  - Do not run the code before writing your expected results.
- Write a testing program that will call your function with the sample data and compare the output with the expected results.
  - You can do this with just the .h files, and link the code in later.

Example Program Design
Writing a Cipher

Exercise

- Design an algorithm to shift encode one character using a character array of size n [1,1000] by an arbitrary value from 0 to n. If a character is not in the array don't change it.
- Example:
  - Array: | A | B | C | D | E |
  - Input 1: shift 'B' by 3 to the right.
  - Output 1:
  - Input 2: shift 'F' by 1 to the left.
  - Output 2:

Exercise (modified)

- Design algorithm to shift encode one character using a character array of size n [1,1000] by an arbitrary value from **-2,147,483,648** to **2,147,483,647**. If a character is not in the array don't change it.
- Example:
  - Array: | A | B | C | D | E |
  - Input 1: shift 'B' by **37** to the right.
  - Output 1:
  - Input 2: shift 'F' by **100** to the left.
  - Output 2:

Design a Shift Cipher interface

- **Part 1 (Simple Shift):** Design a simple shift cipher program that takes 3 arguments a positive integer as the shift amount, an indicator for encryption or decryption, and the input file to be encrypted or decrypted. Your program should be called SimpleShift
- **Part 2 (Variable Shift):** Suppose that you want to use a variable shift for your encryption such that each next character is shifted based on the next number in the deterministic sequence. Your Program should take 2 arguments, the indicator for encryption or decryption, and the input filename to be encrypted or decrypted.

Part 1 (Simple Shift):

- Design a simple shift cipher program that takes 3 arguments a positive integer as the shift amount, an indicator for encryption or decryption, and the input file to be encrypted or decrypted. Your program should be called SimpleShift
- ```
  string simpleShift(int shift, istream input) {
     char c;
     string s;
     while(input >> noskipws >> c) {
       s += shiftChar(c,shift);
     }
     return s;
  }
  ```
  ```