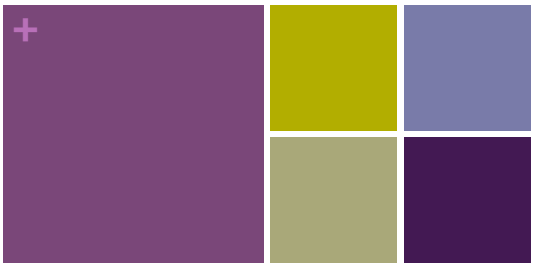

Testing and
Program Design
Ex: Encoded Messages

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.

Which of these functions are stateless?

- double sin(double theta); // computes the sine of theta
- double area(double radius); // computes the area of a circle
- void srand(unsigned int seed); // sets the seed for rand()
- unsigned int rand(); // returns the next random number
- double sum(int* values, int n); // returns the sum of the n values.
- char nextChar(); // returns the next character from a file
- char nextChar(istream s); returns the next character from s

Why should we write stateless functions?

Why should we write stateless functions?

- Self contained

Why should we write stateless functions?

- Self contained
- Deterministic outcome

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

Encoding/Decoding

- Mapping from meaning to code:
  - temporal/regional example:
    - excellent or awesome ->
      - groovy
      - swell
      - hip
      - rad
      - cool
      - tight
      - wicked
      - bad

Encoding/Decoding

- Mapping from meaning to code:
  - the government (or authority) -> The man
  - marriage caused by pregnancy -> shotgun wedding
  - marijuana -> Mary Jane
  - hunger -> crying baby
  - dirty diaper -> crying baby
  - tired -> crying baby
  - pregnant -> bun in the oven
  - pregnant -> knocked up
  - excellent -> bad
  - awful -> bad

Caesar Cipher (Shift Cipher)

- A replacement cipher where each character is shifted by a fixed number.
- IBM is encoded to HAL with a left shift of one character.
- Dad is encode to heh with a right shift of 4
- How do you decode the shift cipher?

Right Shift of Three

A	B	C	D	E	F	G	H	I	J
D	E	F	G	H	I	J	K	L	M

K	L	M	N	O	P	Q	R	S	T
N	O	P	Q	R	S	T	U	V	W

U	V	W	X	Y	Z
X	Y	Z	A	B	C

Alphabet as an Array

A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	5	6	7	8	9

K	L	M	N	O	P	Q	R	S	T
10	11	12	13	14	15	16	17	18	19

U	V	W	X	Y	Z
20	21	22	23	24	25

Alphabet Re-ordered

A	M	X	D	O	Z	G	H	I	R
0	1	2	3	4	5	6	7	8	9

K	L	B	N	E	W	Q	J	S	T
10	11	12	13	14	15	16	17	18	19

P	V	U	C	F	Y
20	21	22	23	24	25

Alphabet Re-ordered With Additional Characters

A	M	X	D	O	Z	G	H	I	R
0	1	2	3	4	5	6	7	8	9

K	L	B	N	E	W	Q	J	S	T
10	11	12	13	14	15	16	17	18	19

P	V	U	C	F	Y	.	:	!	?
20	21	22	23	24	25	26	27	28	29

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.

What functions do we need?

What functions do we need?

- encode
- decode

What extra functions do we need to implement sequential shift amount?

- encode
- decode

What extra functions do we need to implement sequential shift amount?

- encode
- decode
- getShift

What extra functions do we need to implement sequential shift amount?

- encode
- decode
- getShift

- Let's create the prototypes!

Further Reading

- http://www.simonsingh.net/The_Black_Chamber/index.html
- http://www-rohan.sdsu.edu/~gawron/crypto/lectures/shift_cipher.htm
- http://en.wikipedia.org/wiki/Caesar_cipher