pset 2: Crypto

Zamyla Chan | zamyla@cs50.net

pset 2

0. A Section of Questions
1. Caesar
2. Vigenère

Toolbox

- `sudo yum -y update`
- ASCII Chart
- Command Line Arguments
- Arrays
- Strings

ASCII Chart

A	B	C	D	E	F	G	H	I	…
65	66	67	68	69	70	71	72	73	…

a	b	c	d	e	f	g	h	i	…
97	98	99	100	101	102	103	104	105	…

Caesar

k = 0

A → A

B → B

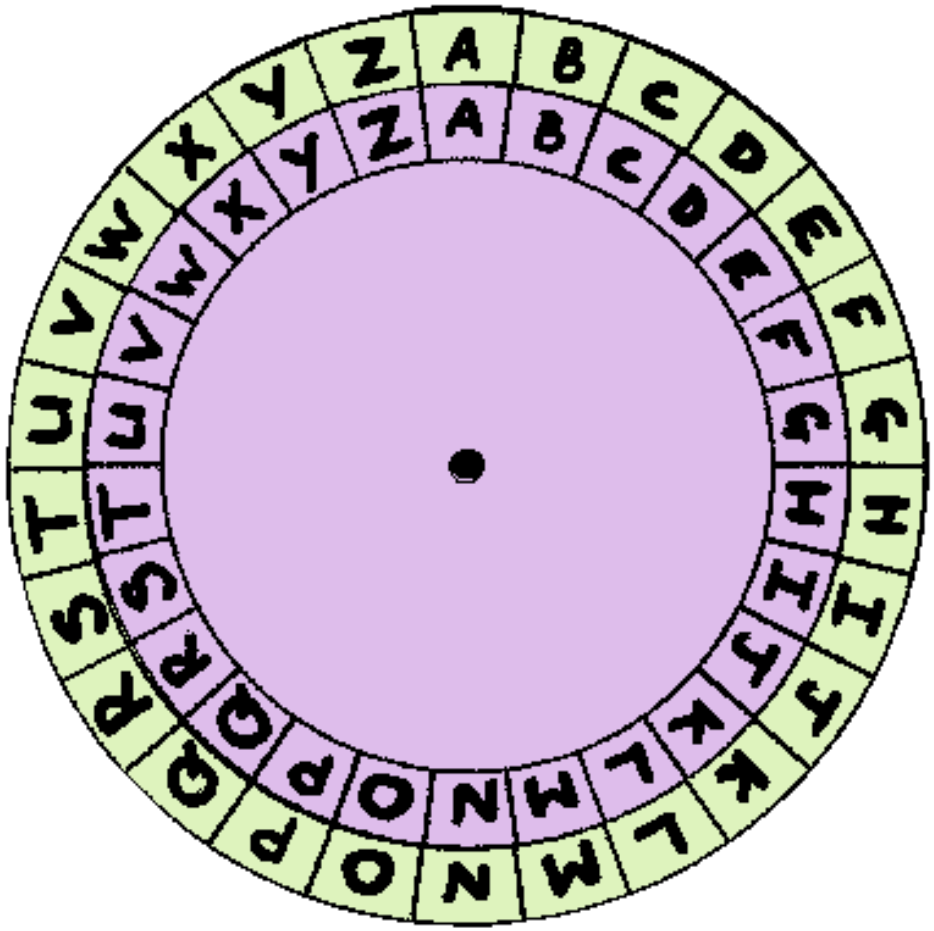
C → C

…

X → X

Y → Y

Z → Z



img src: http://www.maths-resources.net

k = 3

A → D

B → E

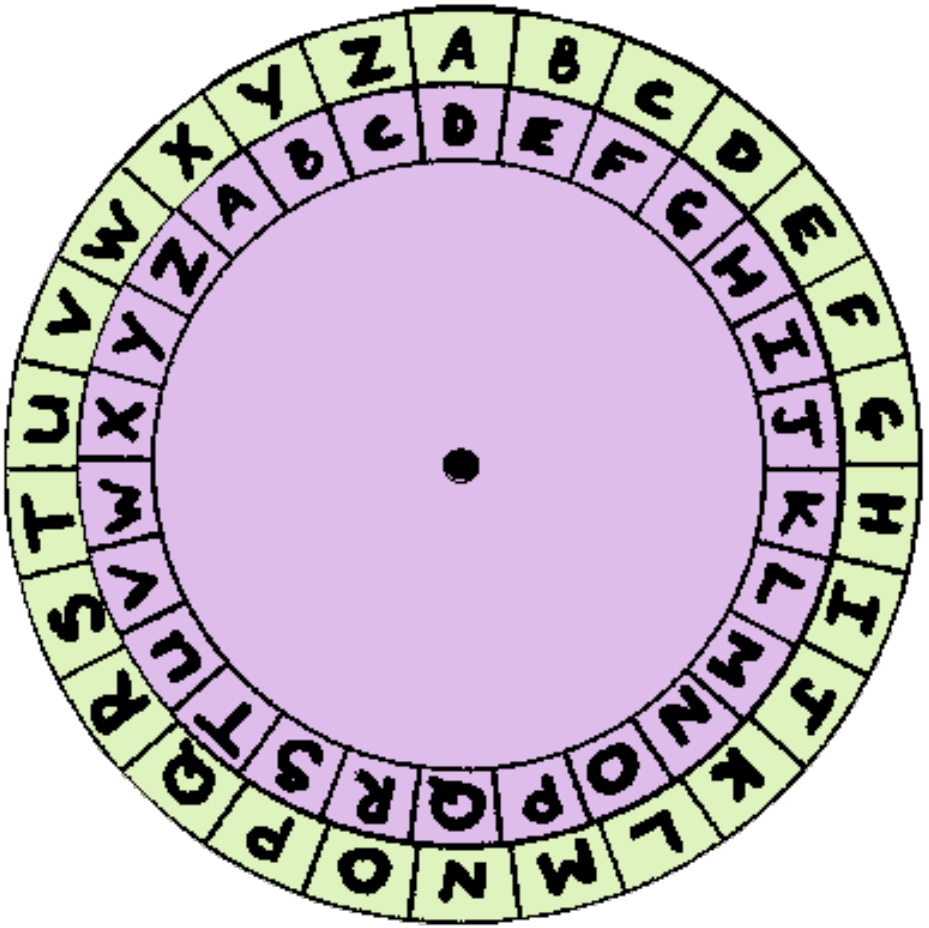
C → F

…

X → A

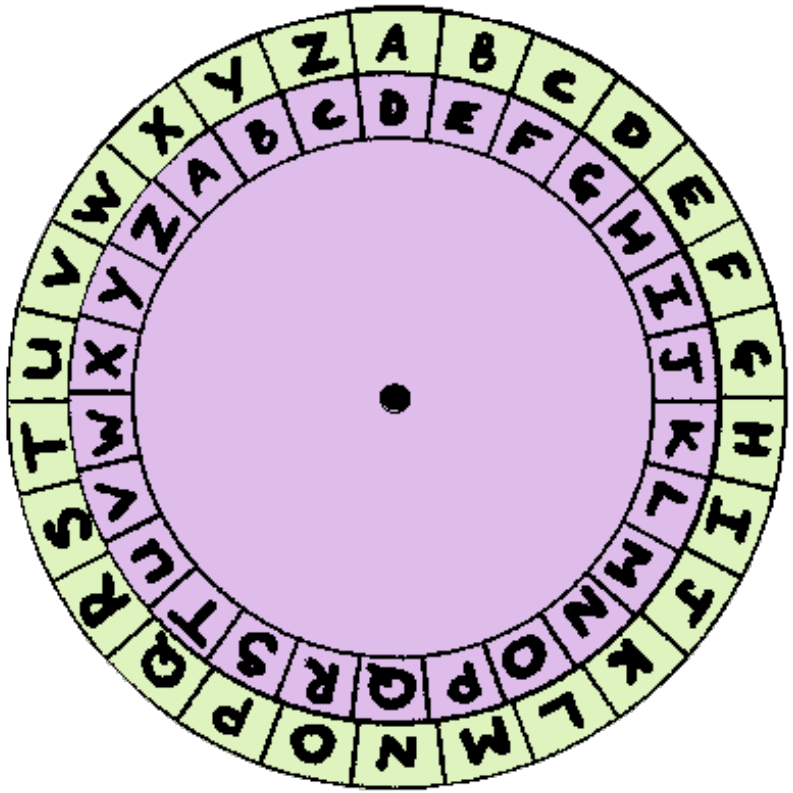
Y → B

Z → C



img src: http://www.maths-resources.net

Example




```
./caesar 3
ABCDEFGHIJKL
DEFGHIJKLMNO

./caesar 3
This is CS50!
Wklv lv FV50!
```

TODO

- Get the key
  - 2nd command line argument

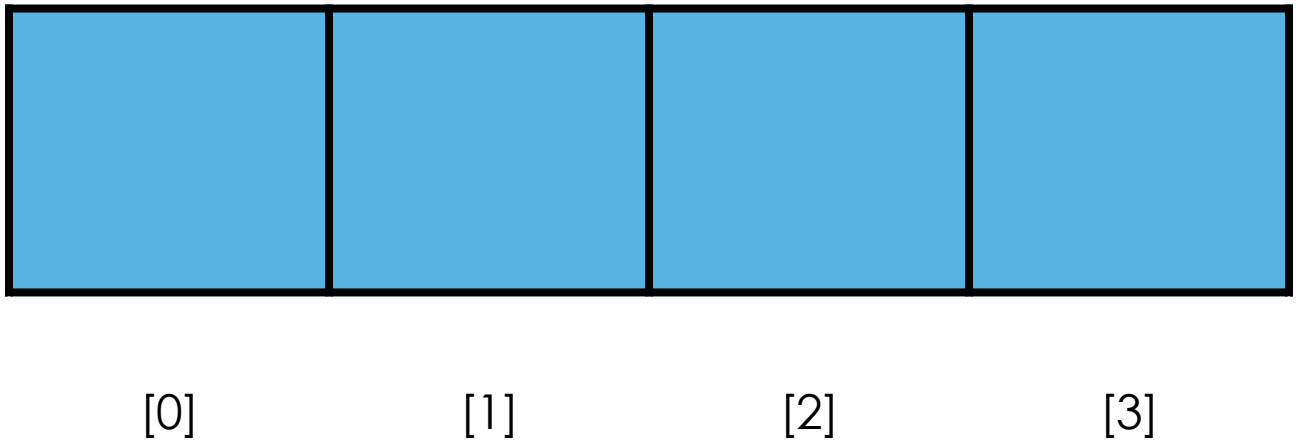
argc, argv

```
int main (int argc, string argv[])
```

- **argc**
  - int
  - the number of arguments passed
- **argv**
  - array of strings
  - the list of arguments passed

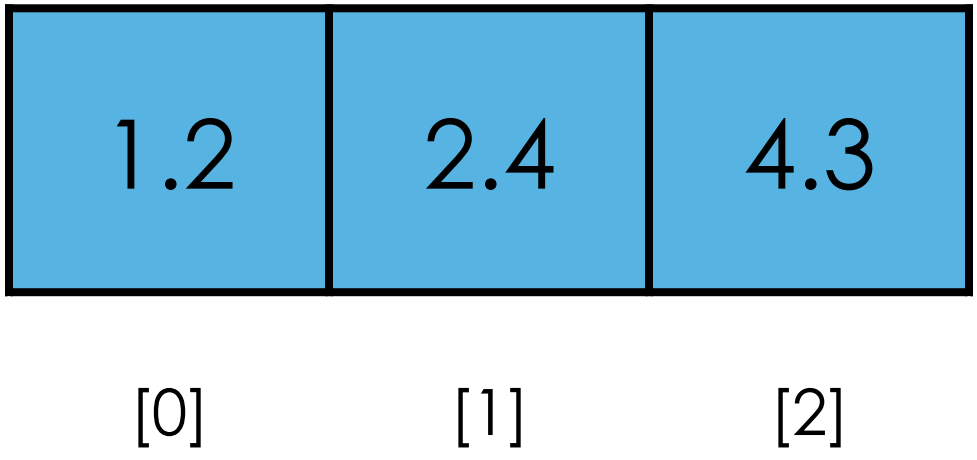
Arrays

- Data structures that hold multiple values of the same type
- Entries are zero-indexed



Creating an Array

```
double mailbox[3];
mailbox[0] = 1.2;
mailbox[1] = 2.4;
mailbox[2] = 4.3;
```



argc, argv

```
./hello David Malan
```

- argc → 3
- argv[0] → "./hello"
- argv[1] → "David"
- argv[2] → "Malan"

args.c

TODO

- Get the key
  - ☑ In the command line argument
  - Check for correct usage
    - They must give you one key! No more, no less.
    - Convert key to an int

atoi

- Converts a string to an integer

```
string num = "50";
int i = atoi(num);
```

i → 50

- Not part of stdio or cs50 libraries...
  - `man atoi`

TODO

- ☑ Get the key
- ☐ Get the plaintext
  - GetString()
  - All user inputted strings are “correct”

TODO

- ☑ Get the key
- ☑ Get the plaintext
- ☐ Encipher

Strings

- A string is just an array of characters

```
string text = "This is CS50";
```

  - text[0] → `T`
  - text[1] → `h`
  - text[2] → `i`
  - etc...

Length of String

```
string text = "This is CS50";
int length = strlen(text);
```

- How would you iterate over each character in a string?
  - strlen.c

k = 6

I	'	m		d	i	z	z	y	!
73	39	109	32	100	105	122	122	121	33

o	'								
79	39								

Reminders

- Only encipher letters
  - non-letters (symbols) remain unchanged
- Preserve capitalization

isalpha, isupper, islower

```
bool alpha = isalpha('T');
```
alpha → true

```
bool upper = isupper('h');
```
upper → false

```
bool lower= islower('i');
```
lower → true

ASCII Math

- Enclose a character with a single quote to use its ASCII number
  - ‘C’ → 67
  - ‘S’ → 83
  - ‘f’ → 102
- You can subtract, add, etc.
  - asciimath.c

k = 6

I	'	m		d	i	z	z	y	!
73	39	109	32	100	105	122	122	121	33

O	'	s		j	o	Ç	Ç		
79	39	115	32	106	111	128	128		

Reminders

- Only encipher letters
  - Non-letters (symbols) remain unchanged
- Preserve capitalization
- Wraparound the alphabet
  - Stay within A-Z and a-z
  - Letters can't become symbols

Modulus %

- x % y gives the remainder of x / y

```
int ans1 = 27 % 15;
```

ans1 → 12

```
int ans2 = 17 % 2;
```

ans2 → 1

Modulus %

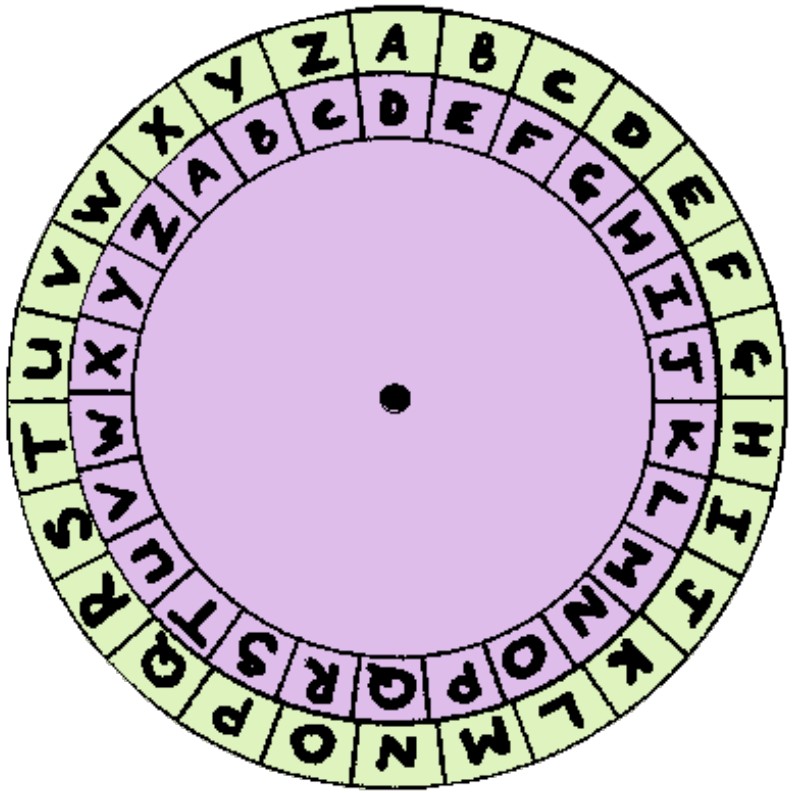
- What's "leftover" after you subtract y from x as many times as possible
- Useful for wrapover. How?

k = 6

I	'	m		d	i	z	z	y	!
73	39	109	32	100	105	122	122	121	33

O	'	s		j	o	f	f	e	!
79	39	115	32	106	111	102	102	101	33

Enciphering



$c_i = (p_i + k) \% 26$

- c_i : i^{th} letter of ciphertext
- p_i : i^{th} letter of plaintext
- k : key
- % 26

‘z’ → ‘f’

ASCII Values

z = 122

f = 102

(‘z’ + 6) % 26

= 24

- ... Should be 102 for ‘f’

Alphabetical Index

z = 25

f = 5

(25 + 6) % 26

= 5

- ... This makes sense, but it still isn’t ‘f’

TODO

- ☑ Get the key
- ☑ Get the plaintext
- ☑ Encipher
- ☐ Print ciphertext

Vigenère

Vigenère

- Get the keyword
- Get the plaintext
- Encipher
- Print ciphertext

Vigenère

- Similar to Caesar! Except the shift is represented by the keyword letters
  - ‘A’ and ‘a’ → 0
  - ‘F’ and ‘f’ → 5
  - ‘Z’ and ‘z’ → 25

$$c_i = (p_i + k_j) \% 26$$

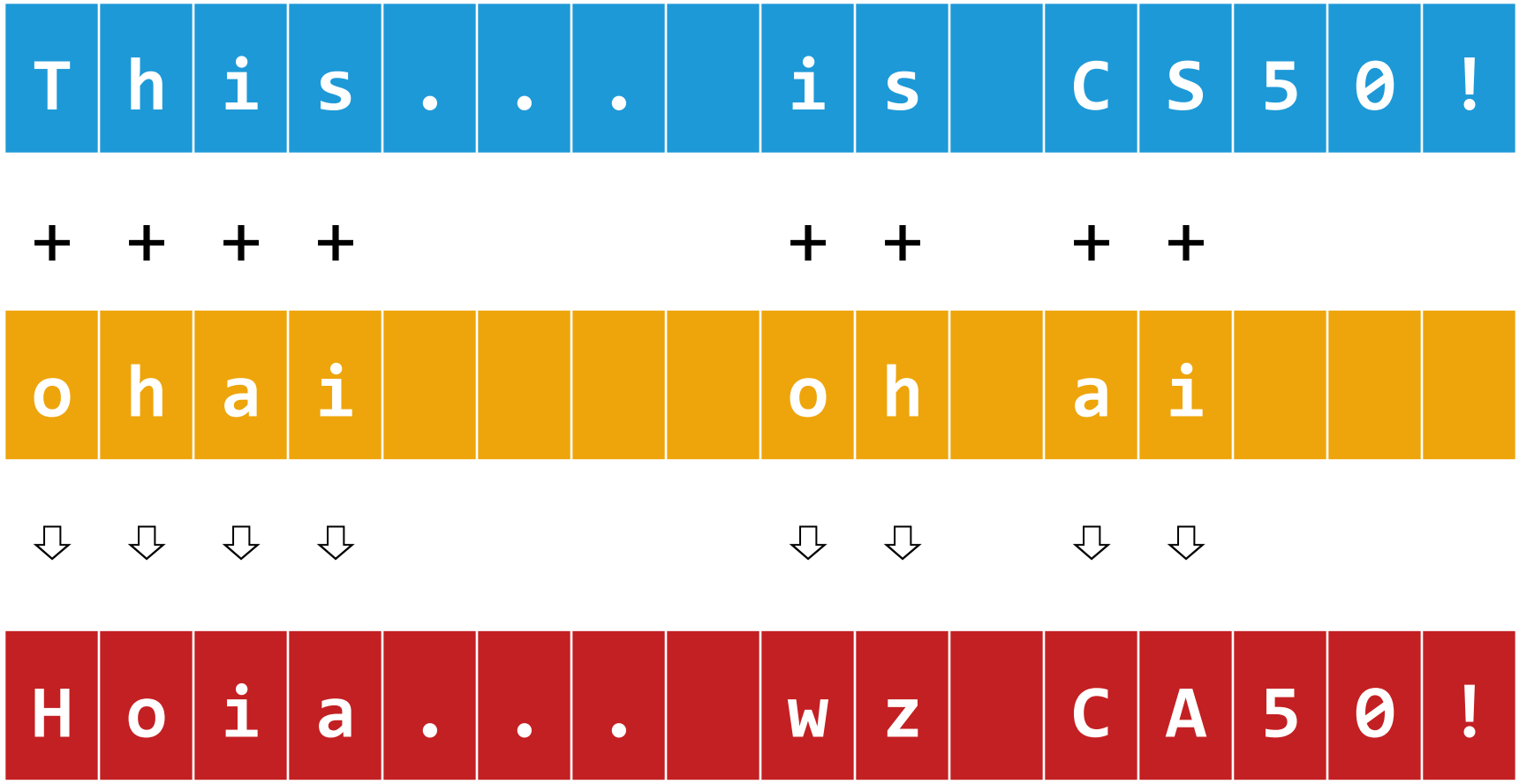
- c_i : i^{th} letter of ciphertext
- p_i : i^{th} letter of plaintext
- k_j : j^{th} letter of the key
- %26

Vigenère

```
jharvard@appliance (~/Dropbox/pset2): ./vigenere ohai
This... is CS50!
Hoia... wz CA50!
```

- keyword in the command line
  - Already a string
- Preserve capitalization
- Only encipher letters

./vigenere ohai



Reminders

- Only advance to the next letter in the keyword if the character in plaintext is a letter
- Wraparound to beginning of keyword when end of keyword
- Keep track of:
  - position in plaintext
  - position in keyword

this was walkthrough 2