# This is Section!

# Agenda

- Who am I and what are we doing here?
- Arrays
- ASCII
- Functions
- Command Line Arguments

# Who am I?

Allison Buchholtz-Au
TF, PAF, best friend, etc
abuchholtzau@college.harvard.edu

# Why are we here?

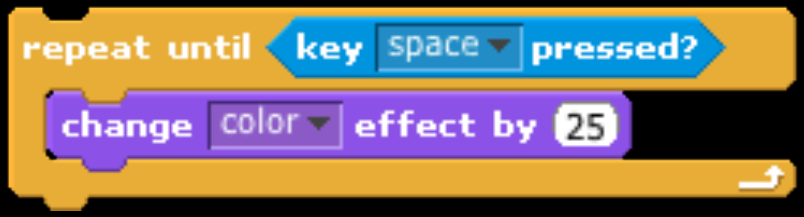
"Sections are a time to dive in and get some hands on experience with topics mentioned in class or in study materials"

# Notes on Section

- Support
- Meet us halfway
- Grading
- Tips

# Loops

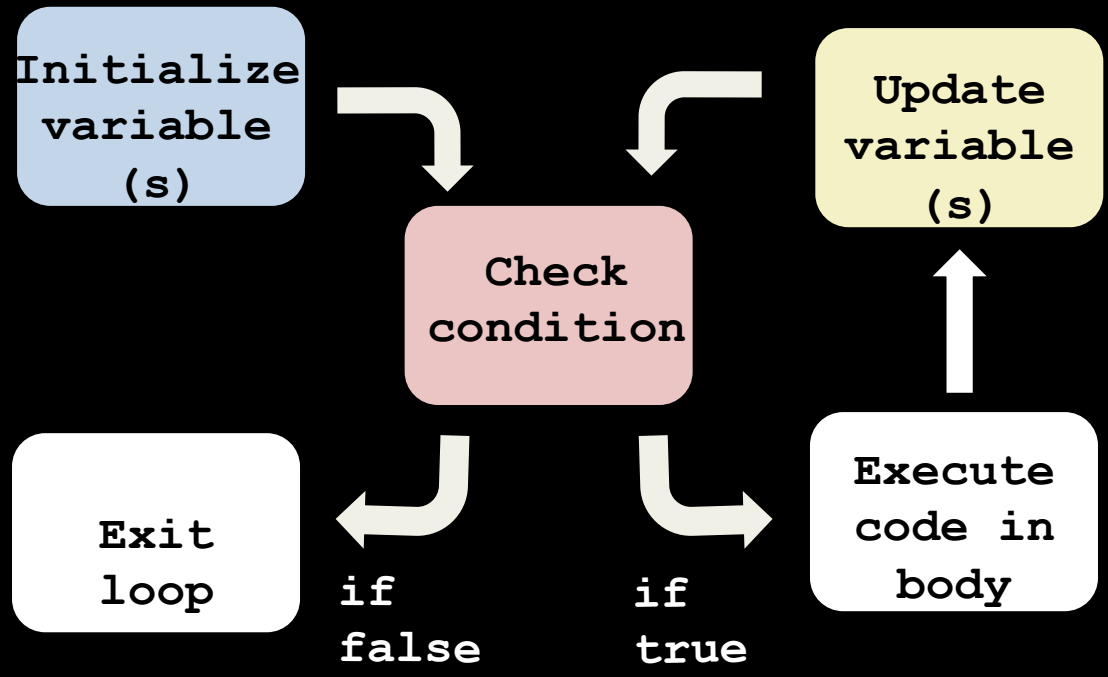

repeat 5
play sound meow


repeat until key space pressed?
change color effect by 25


forever
move 10 steps
if on edge, bounce

# For Loops

```
for (initialization; condition; update)
{
    execute this code
}
```

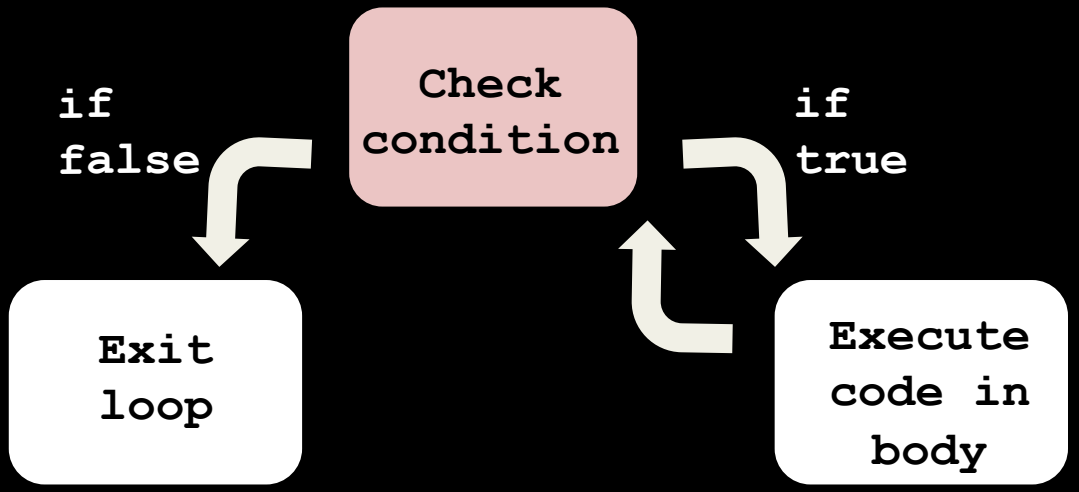


# Example #1
# Prints “This is CS50!” ten times


```
for (int i = 0; i < 10; i++)
{
    printf("This is CS50!\n");
}
```

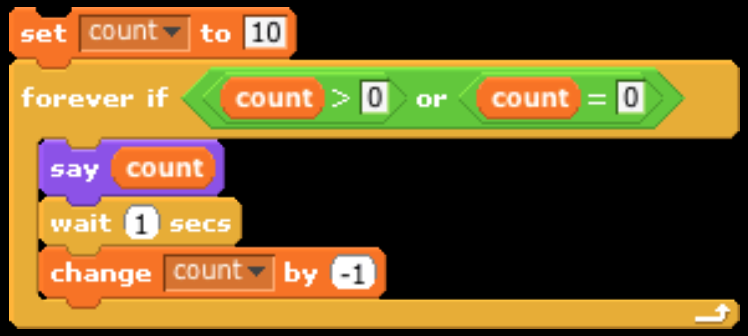
# While Loops

```
while (condition)
{
    execute this code
}
```



# Example

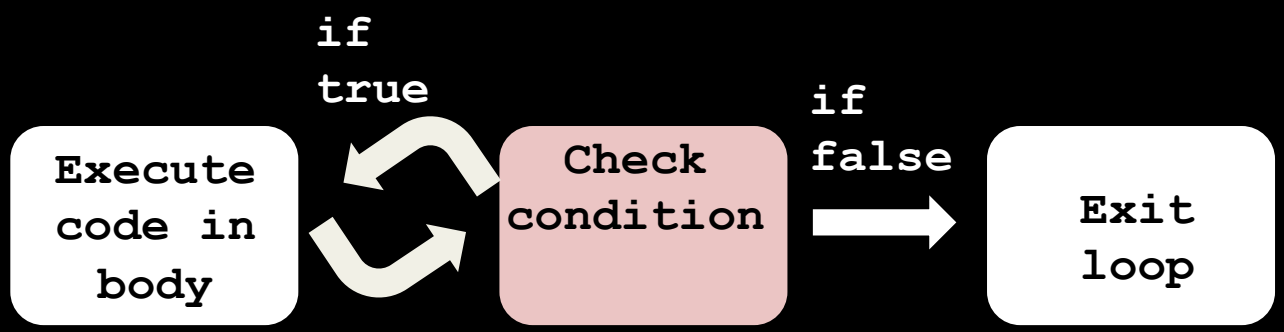
## Counts down from 10 to 0




```
int count = 10;
while (count >= 0)
{
    printf("%i\n", count);
    count--;
}
```

# Do While Loops

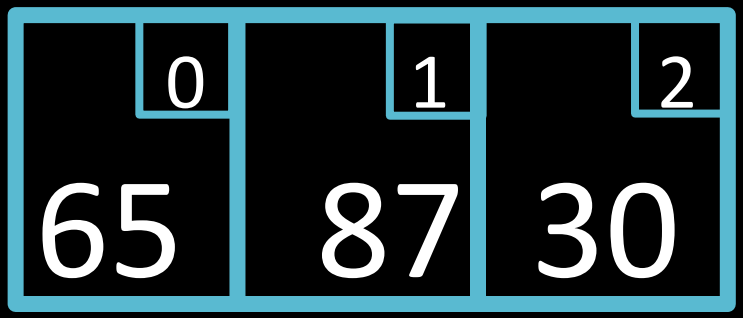
```
do
{
    execute this code
}
while (condition);
```



# Example #5
# Reprompts until user enters a positive number

```
int input;
do
{
    printf("Enter a positive number: ");
    input = GetInt();
}
while (input < 1);
```

# ARRAYS

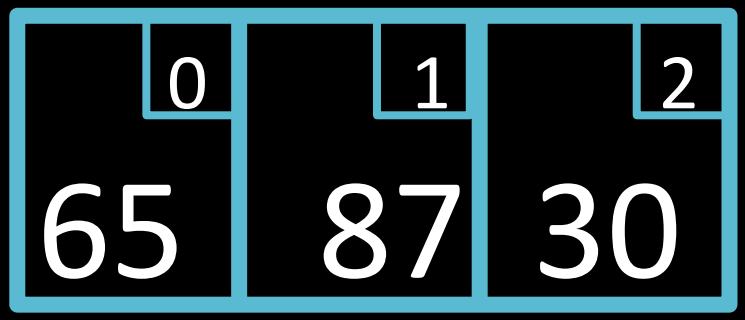


# Creating an array

```
<data type> name[<size>];
```

Example:

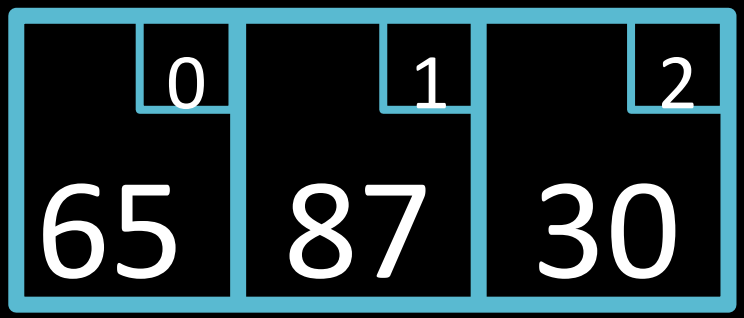
```
int temperature[3];
temperature[0] = 65;
temperature[1] = 87;
temperature[2] = 30;
```




OR

```
int temperature[] = { 65, 87, 30 };
```

# Accessing Elements




```
for (int i = 0; i < 3; i++)
{
      printf("%d\n", temperature[i]);
}
```

```c
#include <stdio.h>
#include <cs50.h>

#define CLASS_SIZE 30

int main(void)
{
      // declare array
      int scores_array[CLASS_SIZE];

      // populate array
      for (int i = 0; i < CLASS_SIZE; i++)
      {
      printf("Enter score for student %d: ", i);
      scores_array[i] = GetInt();
   }
}
```

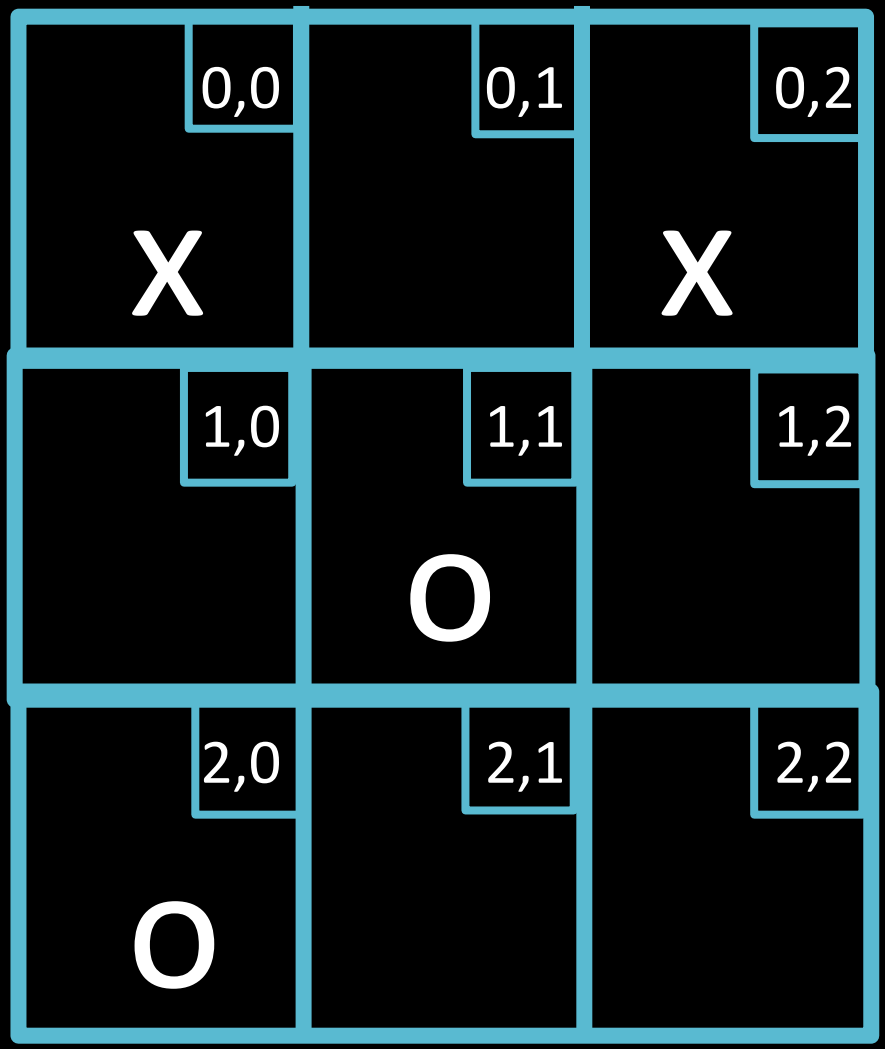
# Where’s the bug?

```
string class[3] = { "Sam", "Jess", "Kim" };

for (int i = 0; i <= 3; i++)
{
      printf("%s\n", class[i]);
}
```

# Multidimensional Arrays

```
char board[3][3];
board[1][1] = 'o';
board[0][0] = 'x';
board[2][0] = 'o';
board[0][2] = 'x';
```



# Example
# Calculates string length

```
string s = GetString();
int length = 0;
while (s[length] != '\0')
    length++;
```

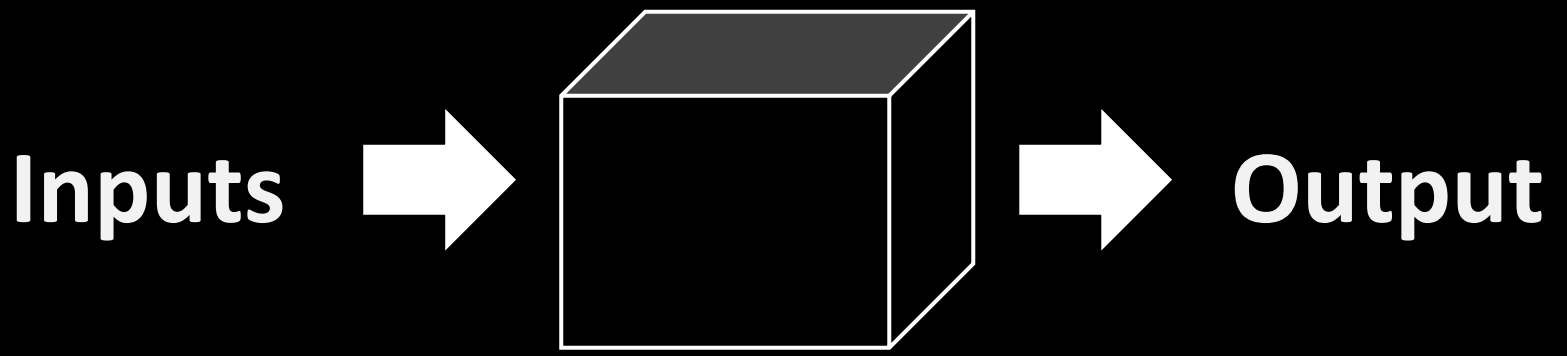
# Example

Create an array with integers 1,2,3 and then print them out

# Example
## Create and array and print out elements

```
int example = { 1, 2, 3 };
for (int i = 0; i < 3; i++)
{
    printf("%i \n”, example [i]);
}
```

# Functions



# Why functions?

- **Organization**
- **Simplification**
- **Reusability**

# Defining a Function

```
int cube(int input)
{
    int output = input * input * input;
    return output;
}
```

Header

function name

return type

parameter list

```
int cube(int input)
{
    int output = input * input * input;
    return output;

}
```

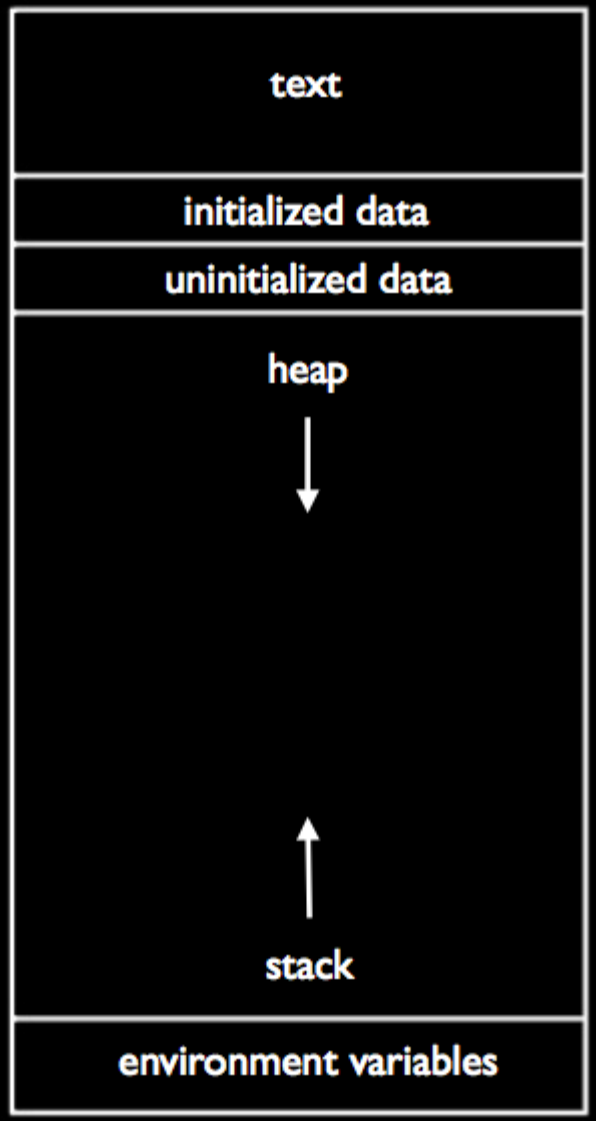
Body

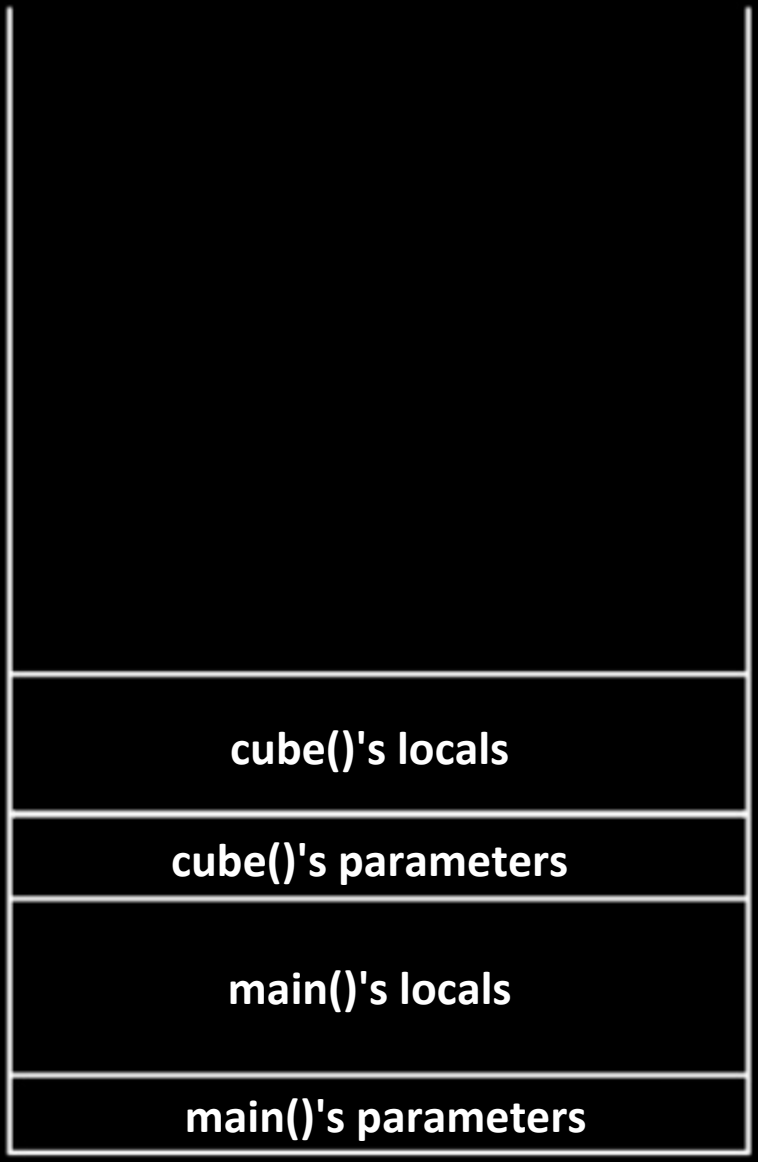
```c
#include <stdio.h>

int cube(int input);

int main(void)
{
    int x = 2;
    printf("x is %i\n", x);
    x = cube(x);
    printf("x is %i\n", x);
}

int cube(int input)
{
    int output = input * input * input;
    return output;
}
```


text
initialized data
uninitialized data
heap
stack
environment variables


cube()'s locals
cube()'s parameters
main()'s locals
main()'s parameters

```c
#include <stdio.h>
void swap(int a, int b);

int main(void)
{
    int x = 1, y = 2;
    swap(x, y);
    printf("x is %i\n", x);
    printf("y is %i\n", y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

# Command-line Arguments

```
int main(void)

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

# Test Yourself

```
jharvard@appliance (~): ./copy infile outfile
```

1. What is argc?
2. What is argv[0]?
3. What is argv[1]?
4. What is argv[2]?
5. What is argv[3]?
6. What is argv[4]?

# PSet Review!

- Review of ASCII
- Conversion of command line inputs
- Modulo

# ASCII maps characters to numbers

| INT | CHAR | | INT | CHAR | INT | CHAR | INT | CHAR |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | (null) | 32 | SPACE | 64 | @ | 96 | ` |
| 1 | SOH | (start of heading) | 33 | ! | 65 | A | 97 | a |
| 2 | STX | (start of text) | 34 | " | 66 | B | 98 | b |
| 3 | ETX | (end of text) | 35 | # | 67 | C | 99 | c |
| 4 | EOT | (end of transmission) | 36 | $ | 68 | D | 100 | d |
| 5 | ENQ | (enquiry) | 37 | % | 69 | E | 101 | e |
| 6 | ACK | (acknowledge) | 38 | & | 70 | F | 102 | f |
| 7 | BEL | (bell) | 39 | ' | 71 | G | 103 | g |
| 8 | BS | (backspace) | 40 | ( | 72 | H | 104 | h |
| 9 | HT | (horizontal tab) | 41 | ) | 73 | I | 105 | i |
| 10 | LF | (line feed) | 42 | * | 74 | J | 106 | j |
| 11 | VT | (vertical tab) | 43 | + | 75 | K | 107 | k |
| 12 | FF | (form feed) | 44 | , | 76 | L | 108 | l |
| 13 | CR | (carriage return) | 45 | - | 77 | M | 109 | m |
| 14 | SO | (shift out) | 46 | . | 78 | N | 110 | n |
| 15 | SI | (shift in) | 47 | / | 79 | O | 111 | o |
| 16 | DLE | (data link escape) | 48 | 0 | 80 | P | 112 | p |
| 17 | DC1 | (device control 1) | 49 | 1 | 81 | Q | 113 | q |
| 18 | DC2 | (device control 2) | 50 | 2 | 82 | R | 114 | r |
| 19 | DC3 | (device control 3) | 51 | 3 | 83 | S | 115 | s |
| 20 | DC4 | (device control 4) | 52 | 4 | 84 | T | 116 | t |
| 21 | NAK | (negative acknowledge) | 53 | 5 | 85 | U | 117 | u |
| 22 | SYN | (synchronous idle) | 54 | 6 | 86 | V | 118 | v |
| 23 | ETB | (end of transmission block) | 55 | 7 | 87 | W | 119 | w |
| 24 | CAN | (cancel) | 56 | 8 | 88 | X | 120 | x |
| 25 | EM | (end of medium) | 57 | 9 | 89 | Y | 121 | y |
| 26 | SUB | (substitute) | 58 | : | 90 | Z | 122 | z |
| 27 | ESC | (escape) | 59 | ; | 91 | [ | 123 | { |
| 28 | FS | (file separator) | 60 | < | 92 | \ | 124 | \| |
| 29 | GS | (group separator) | 61 | = | 93 | ] | 125 | } |
| 30 | RS | (record separator) | 62 | > | 94 | ^ | 126 | ~ |
| 31 | US | (unit separator) | 63 | ? | 95 | _ | 127 | DEL |

# ASCII Math

**What will print?**

```
printf("%d\n", 'a' - 'A');
printf("%c\n", 'B' + ('a' - 'A'));
printf("%c\n", 'b' - ('a' - 'A'));
printf("%c\n", 'B' + 1);
printf("%c\n", ('z' - 'a' + 1) % 26 + 'a');
```

# atoi()

- Converts a string to an int
- argv will be a string so we need to change it to an int!

- Not necessary for Vigenere…

# Modulo : %

- What if we are given really large number for Caesar?
- What happens when we reach the end of the string in Vigenere?

Modulo gives you the remainder of the division of the first number by the second!

# Examples

1. `55 % 10`
2. `3 % 5`
3. `8 % 8`
4. `16 % 15`
5. `(1 + 2) * 2 % 2 + 2`