Programming in C

Today, you'll begin your very first program in C!
- You already know how to program in C++! 🎉
- Programming in C is a simplification of the C++ programming.

1. Program Starting Point of ALL C PROGRAMS:

2. Printing Using `printf()` (and include `<stdio.h>`):

```c
3/printf.c

int main() {
    int i = 42;
    char *s = "Hello, world!";
    float f = 3.14;

    printf("%d  %s  %f\n", i, s, f);
    printf("%d\n", s[0]);
    printf("%d\n", s);
    printf("%d\n", f);
    return 0;
}
```

3. Pointers:

4. Heap Memory Allocation:

```c
3/malloc.c

int main() {
    char *s = malloc(10);
    int *num = malloc(sizeof(int));
    printf("%p %p\n", s, num);
    return 0;
}
```

5. Strings – There is no “data type” in C known as a string. Instead, we refer to “C Strings” as a sequence of characters:
- A ‘C string’ is just a character pointer: __________.
- The string continues until it reaches a __________ byte.
- C will automatically include the NULL byte ONLY when using double quotes in your code (not counted as part of the length, but does require memory – extremely tricky!)

```c
3/string.c

char *s = malloc(6);
strcpy(s, "cs340");
printf("%s %s %s %s\n", s, s[0], s[0], s[0]);
printf("%s == "%s", strlen(s): %ld\n\n", s, strlen(s));
char *s2 = s + 2;
printf("%s %s %s %s\n", s, s1, s2, s3);
printf("%s == "%s", strlen(s2): %ld\n\n", s2, strlen(s2));
*s2 = 0;
printf("%s %s %s %s\n", s, s1, s2, s3);
printf("%s == "%s", strlen(s): %ld\n\n", s, strlen(s));
```

```c
3/utf8.c

char *s = malloc(5);
s[0]=0xF0; s[1]=0x9F; s[2]=0x8E; s[3]=0x89; s[4]=0x00;
char *s1 = \xF0\x9F\x8E\x89\nchar *s2 = "🎉\nchar *s3 = \U0001f389\n// \U - must be 8 bytes
```

Some extremely useful built in string functions:
- `strcmp(char *s1, char *s2)` -- Compares two strings
- `strcat(char *dest, char *src)` -- Concatenate two strings
- `strcpy(char *dest, char *src)` -- Copies a string
- `strlen(char *s)` -- Returns the length of the string
Logic Gates and Truth Tables
We can begin to define the building blocks of the CPU by basic instructions with input bits and output bits through logical gates.

- By convention, you will see that the input bits are labeled A and B by default.

Logic Gate #1:

Logic Gate #2:

Logic Gate #3:

Logic Gate Challenge: A XOR B

We can also express this in a table known as a truth table:

<table>
<thead>
<tr>
<th>Op.</th>
<th>Binary</th>
<th>Math</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>x</td>
<td>1100</td>
<td>110011 101</td>
</tr>
<tr>
<td>B</td>
<td>y</td>
<td>1010</td>
<td>11 010</td>
</tr>
<tr>
<td>AND</td>
<td>A &amp; B</td>
<td>xy</td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>A</td>
<td>B</td>
<td>x + y</td>
</tr>
<tr>
<td>XOR</td>
<td>A ^ B</td>
<td>x XOR y</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>!A</td>
<td>x’</td>
<td></td>
</tr>
</tbody>
</table>

Truth Table: Half Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A + B</th>
<th>SUM</th>
<th>CARRY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Truth Table: Half Adder

Truth Table for a Full Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>CARRY_{in}</th>
<th>SUM</th>
<th>CARRY_{out}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Circuit Diagram for a “Full Adder”:

Full Adder:

Chaining Circuits Together: _______________________

Disadvantages: