CS50 Section. Week 3. 9/22/15.

Neel Mehta

About me

Neel Mehta '18
neelmehta@college.harvard.edu
(215) 990-6434

Resources

- C language reference: https://reference.cs50.net/
- CS50 Study: https://study.cs50.net/
- CS50 Discuss: https://cs50.harvard.edu/discuss
- CS50 Style Guide: https://manual.cs50.net/style/
- Handouts and practice problems: https://github.com/hathix/cs50-section

Grading

Psets are graded along the axes of:

- **Scope: did you do every part of the pset?** (This is pretty easy to get full points on.)
- **Correctness: does your code work?** We run tests on your programs and make sure they produce the desired output. Running `check50` is a great start.
- **Design: did you solve the problem effectively?** This is subjective, but it's all about producing efficient, well-designed code. Ask me if you need pointers.
- **Style: is your code well-formatted?** These should be easy points -- just make sure your programs pass `style50`.

When I'm grading your psets, I'll focus more on qualitative feedback (comments) -- those matter more than the numerical score you get. Using my comments will probably be the best way for you to improve over the semester, which is what I care the most about.

General pset tips

`style50` will make sure your style is OK. It's great at catching dumb syntax mistakes, so use it whenever you get strange compilation errors. And, of course, use it before you submit!

```
style50 my-file.c
```

`check50` will run tests to make sure your program gives the expected output when given some input. Make sure your program passes all of its tests before you submit! Most psets have a `check50`; check the pset spec to find them!

Generally, here's how I advise you approach psets:

- The algorithms -- the logic -- are the hardest parts of the pset. For pset1, it was figuring out how many "#"'s and spaces to print on any line. For pset2, it's figuring out how to rotate the text properly. Work out your logic on paper before you

even touch your computer.
- Find a pset partner. Just talking to someone makes your thinking much clearer. It's also nice to have someone to spend those long pset hours with. Make friends in section, office hours, your entryway, etc. Or just tell me and I can put you in touch with someone!
- Go to office hours early in the week: Monday or Tuesday. You'll encounter smaller crowds and get more time from TFs.
- Start the psets well before you come to section so that you have specific, actionable questions I can help you with.

pset2 tips

Miscellaneous tips

- Make sure you check if `getString` returns `NULL`.
- Use `printf` to print error messages if the user doesn't give the right command-line input.
- When encrypting text for Caesar and Vigenere, use modulo (`%`) to make sure your letters wrap around from, e.g., Z to A.

The hardest parts of this pset

- Getting your letter-rotating algorithm right. Work out the math on paper before you start coding.
- All the little things: validating input, making sure you don't miss anything in your `for` loops, etc.

`for` Loops

`for` loops are great for repeating blocks of code or for iterating over every element in an array.

Here's an example of a for loop:

```
// prints 0, 1, ... 9
for (int i = 0; i < 10; i++)
{
    printf("%i\n", i);
}
```

And a more advanced one:

```
// prints 10, 15, ... 50
for (int j = 10; j <= 50; j += 5)
{
    printf("%i\n", j);
}
```

Arrays

Arrays let you store a bunch of related pieces of data in a clean, organized way.

Here's the primary way to create an array:

```
// <datatype> <name>[<size>];
string my_classes[4];
// you don't have to fill in all the elements of the array
my_classes[0] = "CS50";
my_classes[1] = "Ec 10";
my_classes[2] = "Expos 20";
my_classes[3] = "Math 55";
```

You can use this shorthand if you know all the elements of the array ahead of time. Notice that C will infer the array size for you, so you don't need to write it!

```
// <datatype> <name>[] = {elements};
int important_years[] = {1636, 1776, 2015};
```

You can access and change the elements of an array like this:

```
// prints "This is CS50." (without quotes)
printf("This is %s.\n", my_classes[0]);

// prints 1787
important_years[1] = 1787;
printf("%i\n", important_years[1]);
```

Strings

Strings of text are really just arrays of characters. You can work with them just like normal arrays, for the most part:

```
// prints "u" (without quotes)
string band = "Young the Giant";
char third_letter = band[2];
printf("%c\n", third_letter);
```

Command-line arguments

You know how you normally write `./program` to run a program? You can write more, e.g. `./program donuts`, to pass information ("arguments") into the program. Your program can then use these arguments!

```
int main(int argc, string argv[])
{
    // if the user types `./program donuts`,
    // this will print "donuts" (without quotes)
    printf("%s\n", argv[1]);
}
```

Notice that you'll need to change the *header* of `main` to include `int argc, string argv[]`. And notice that the arguments you give start at index 1, because index 0 is the program's name itself!

Challenges

These are designed to be difficult. Work on these with a partner. Assume all necessary header files have been `#include`'d.

Solutions at https://github.com/hathix/cs50-section/tree/master/code/3.

Loops

Print every multiple of 3 from 9 to 18 inclusive.

```
int main(void)
{









}
```

Arrays

Print the last 3 elements of the given array. Use `num_primes` somehow. Should print 7, 11, and 13.

```
int main(void)
{
    int primes[] = {2, 3, 5, 7, 11, 13};
    int num_primes = 6;









}
```

Strings

Print the given string, but reversed. Should print "dravraH".

```
int main(void)
{
    string college = "Harvard";
    int len = strlen(college);









}
```

Command-line arguments

Print the given string the given number of times. Assume that the user gives you the right number of command-line arguments.

For instance, if you run `./cla-challenge Hello 3`, this program should output:

> *Hello*
> *Hello*
> *Hello*

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









}
```

Hint: `atoi` *is your friend. Look it up at https://reference.cs50.net/.*