15-122: Principles of Imperative Computation

Recitation 18

Nivedita Chopra

Today, we'll be writing some C code :)

Hopefully, you have your laptop with you. If not, please sit next to someone with a laptop, so that you can work with them.

Throughout this handout, please refer to the tutorial at this link `http://c0.typesafety.net/tutorial/From-C0-to-C:-Basics.html` (an electronic version of this handout should be up on Piazza, so you won't need to type in the url)

All C0 files (and some library files) mentioned in this handout can be found at `http://www.andrew.cmu.edu/~niveditc/pages/C0toC.html`

Hello World!

The first program to try out in any programming language is one that prints out "Hello World!" (or any other string of your choice), so that you know that you are able to compile and run programs.

Create a file called `hello.c` and write some valid C code that prints out the string "Hello World!" You can refer to the file `hello.c0` for the equivalent C0 code. Also, you'll need to use the `stdio` library for printing.

Now compile the file with
`gcc -Wall -Wextra -Werror -std=c99 -pedantic hello.c`

Fix any compile errors and then run the code using
`./a.out`

This should print out
`Hello World!`

Fun with Strings

Given a string e.g. "Nivedita", we'd like to print out the following:

```
8. Nivedita
7. ivedita
6. vedita
5. edita
4. dita
3. ita
2. ta
1. a
```

Create a file `stackfun.c` and write a function `print_substrings` that takes a string as an argument and prints it out in the form shown above.

Then compile and run the program (don't forget to include a `main` function in the file). The equivalent C0 code is in the `stackfun.c0` file.

Roll Call

We're checking on the attendance records of various students. Each student is represented by a struct with fields for his/her Andrew ID, the number of days he/she was supposed to have attended class, and boolean array with the letters repesenting whether the student was present or absent on a given day. Look at `rollcall.c0` for the C0 version of the code this struct.

- Copy the xalloc.h and xalloc.c file from the website into your working directory. Remember to use xmalloc rather than malloc in your code.
- In a new file, `rollcall.c`, write a function `student_new` that takes in an andrew id and a string of letters (either 'T' or 'F'). The function creates a new student struct, assigns the appropriate values and returns the newly created struct.
- Write a function `count_present` to count the number of days that a given student was present.
- Translate the main function in `rollcall.c0` to valid C code in `rollcall.c`
- Compile and run the code in `rollcall.c` and check that your output matches the following:
```
niveditc :  3/5
medee :  3/3
shayaks :  3/4
sj1 :  3/5
hbovik :  0/0
```
- However, whenever you malloc, you need to free the memory as well. Attempt to free the memory that you have allocated. Remember to free the arrays allocated within each struct! Consider writing a `student_free` function and calling it on each student.
- Test if you have corrrectly freed the memory by compiling the file and then running with Valgrind: `valgrind ./a.out`
- If you see "All heap blocks were freed – no leaks are possible," then you've freed everything that you allocated :)
- Else, the "HEAP SUMMARY" and "LEAK SUMMARY" probably show that some bytes of memory are lost. Most probably, you're not either freeing the student struct correctly or you're not freeing the array of students.