

Homework #1 – From C to Binary

Due Wednesday, Feb 3 at 5:00pm

You must do all work individually, and you must submit your work electronically via Sakai.

All submitted code will be tested for suspicious similarities to other code, and the test will uncover cheating, even if it is “hidden” (by reordering code, by renaming variables, etc.).

Representing Datatypes in Binary

- 1) [5 points] Convert $+49_{10}$ to 8-bit 2s complement integer representation. Show your work!
- 2) [5] Convert -23_{10} to 8-bit 2s complement integer representation. Show your work!
- 3) [5] Convert $+49.0_{10}$ to 32-bit IEEE floating point representation. Show your work!
- 4) [5] Convert -1.25_{10} to 32-bit IEEE floating point representation. Show your work!
- 5) [5] Represent the string “2016: Duke>UNC” (not including the quotes) in ASCII. Use hexadecimal format for representing the ASCII. (Don’t need to show work.)
- 6) [5] Give an example of a number that cannot be represented by a 32-bit computer. Explain your answer.

Memory as an Array of Bytes

Use the following C code for the next few questions.

```

float* e_ptr;

int main() {
    float a = 1.2;
    e_ptr = &a;
    float* b_ptr = malloc (2*sizeof(float));
    b_ptr[0] = 7.0;
    b_ptr[1] = 4.0;
    float c = foo(e_ptr, b_ptr, b_ptr+1);
    free b_ptr;
    if (c > 10.5){
        return 0;
    } else {
        return 1;
    }
}

float foo(float* x, float *y, float* z){
    if (*x > *y + *z){
        return *x;
    } else {
        return *y+*z;
    }
}

```

- 7) [5] Where do the following variables live (global data, stack, or heap)?
- a. a
 - b. b_ptr
 - c. *b_ptr
 - d. *e_ptr
 - e. b_ptr[0]
- 8) [5] What is the value returned by main()?

Compiling C Code

- 9) [10] A high level program can be translated by a compiler (and assembled) into any number of different, but functionally equivalent, machine language programs. (A simplistic and not particularly insightful example of this is that we can take the high level code $C=A+B$ and represent it with either `add C, A, B` or `add C, B, A`.) When you compile a program, you can tell the compiler how much effort it should put into trying to create code that will run faster. If you

type `g++ -O0 -o myProgramUnopt prog.c`, you'll get unoptimized code. If you type `g++ -O3 -o myProgramOpt prog.c`, you'll get highly optimized code. Please perform this experiment on the program `prog.c` located in the "Resources" section of the course Sakai site (in the folder named "Homeworks"). Compile it both with and without optimizations. **Compare the runtimes of each and write what you observe.** (To time a program on a Unix machine, type `"time ./myProgram"`, and then look at the number before the "u". This number represents the time spent executing user code.)

Writing C Code

In the next three problems, you'll be writing C code. You will need to learn how to write C code that:

- Reads in a command line argument (in this case, that argument is "statsfile.txt" without the quotes),
- Opens a file, and
- Reads lines from a file

You may want to consult the internet for help on this. One (of many!) examples of helpful information is at http://www.phanderson.com/files/file_read.html.

You may not use/borrow any code from any external source (internet, textbook, etc.). Plagiarism of code will be treated as academic misconduct.

Your programs must run correctly on the Linux machines in the login.oit.duke.edu cluster. If your program name is `myprog.c`, then we should be able to compile it with: `g++ -o myprog myprog.c`.

If your program compiles and runs correctly on some other machine but not on the Linux machines in the login.oit.duke.edu cluster, the TA grading it will assume it is broken and deduct points. It is your job to make sure that it compiles and runs on the login.oit.duke.edu machines. Code that does not compile or that immediately fails (e.g., with a seg fault) will receive approximately zero points – it is NOT the job of the grader to figure out how to get your code to compile or run.

- 10) [10] Write a C program called `palindromes.c` that prints out the first N positive numbers that are at least two digits and palindromes (i.e., are the same if read left-to-right or right-to-left), where N is an integer that is input to the program on the command line. If your binary executable is called `palindromes`, then you'd run it on an input of 47 with: `./palindromes 47`

You will upload `palindromes.c` into Sakai (via Assignments).

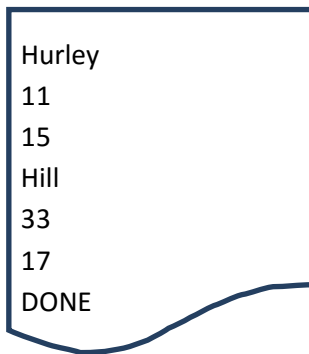
- 11) [20] Write a C program called recurse.c that computes and prints out $f(N)$, where N is an integer greater than zero that is input to the program on the command line. $f(N) = 3*(N+1)+f(N-1)-1$. The base case is $f(0)=1$. **Your code must be recursive.**

The key aspect of this program is to teach you how to use recursion → code that is not recursive, even if it gets the right answer, will be severely penalized!

You will upload recurse.c into Sakai.

- 12) [50] Write a C program called HoopsStats that takes a file as an input (./HoopsStats statsfile.txt). The file is in the following format. The file is a series of player stats, where each player entry is 3 lines long. The first line is the player's last name, the second line is his jersey number (an int), and the third line is his points per game (rounded to an int). After the last player in the list, the last line of the file is the string "DONE". For example:

statsfile.txt



```
Hurley
11
15
Hill
33
17
DONE
```

Your program should output a number of lines equal to the number of players, and each line is the player's name and the difference between his jersey number and his points per game (e.g., $11-15=-4$ for Hurley, $33-17=16$ for Hill). We'll call that stat his JPG. The lines should be sorted in ascending order of JPG, and you must write your own sorting function (you can't just use the qsort library function). For example, the output for the sample statsfile above should be:

```
Hurley -4
Hill 16
```

You must use dynamic allocation/deallocation of memory, and points will be deducted for improper memory management (e.g., never deallocating any memory that you allocated).

Note also that you must use malloc() for dynamic allocation of memory. It may be tempting to take advantage of the compiler's ability to understand an array declaration with a variable size (e.g., `int myArray [NUMPLAYERS]`), but we will deduct points for this. The point of this exercise is to teach you how to manage memory and use malloc().

You will upload HoopStats.c into Sakai.