ECE/CS 250 – Recitation #2 – Prof. Sorin
Getting Started in C Programming

Objective: In this recitation, you will learn how to write, compile, and run simple C programs.

Complete as much of this as you can during recitation. If you run out of time, please complete the rest at home.

1. Task 1: Hello World

As a warm-up, your first task is to write a program that prints out "Hello World" or some other string of your choice. ("ECE/CS 250 is awesome" is another fine example.) First login to a Unix machine (login.oit.duke.edu) like you did in Recitation #1. Use the text editor of your choice to create a file called `hello.c` and then compile it with g++ and run it (as shown below this paragraph). The first line compiles `hello.c` into an executable program called `hello`, and the second line runs the program `hello`. The "-o hello" part of the first line tells g++ to create an executable called `hello`. By default, g++ would've otherwise created an executable called `a.out`. In the second line, you may wonder what the deal is with the "./" – that tells the terminal to look in the current directory for the file to run, which is often necessary for running a program, but not necessary for reading it or moving it or renaming it, etc. (Your current directory can be referred to with "." and its parent directory can be referred to with ".." So if you type "`cd ..`" that'll take you to the parent directory.)

```
g++ -o hello hello.c
./hello
```

2. Task 2: Basic I/O and Type Casts

Write a program that asks the user to input the name of his/her favorite Duke basketball player, the player's height in inches (this height should be an integer), and the player's average number of points scored per game (this last input should be an integer). The program should then output the player's name followed by "scored an average of X points per inch", where X is the average points divided by the height in inches. IMPORTANT: X must be a floating point number, which means you must do some type casting to compute it.

3. Task 3: Functions and Structs

Write a program that uses a two-element struct called HoopsPlayer. This struct has two elements: an int for the player's number, and a float for his average points per game. Write a program that in main() loops and, in each loop, asks the user to input the info for one player (ask for an int then ask for a float). If the user inputs "-1" for the player number, then the loop ends (without asking for a float). For simplicity, you may assume you'll get at most 10 players. After the loop ends, main() must call a

function called sortList(<args>) that prints the list of players sorted in ascending order of points per game. IMPORTANT: for sortList() to work, you have to figure out how to have sortList() get access to the list. A simple way is to declare the list as a global variable. More sophisticated programmers can pass a pointer instead.