Compiling, Gradebot and Homework 3

You are to submit your programs to Gradebot before uploading them to SmartSite. Gradebot will grade their correctness. What follows assumes that you have your Gradebot keys set up properly; if not, see the handout **All About Homework** to learn how to do this.

1 Problem 6

Problem 6 asks you to write a function to read an integer; the interface is documents in the assignment.

To help you, we have placed the input for Gradebot in the directory */home/bishop/ecs30/hw3-6-input* in the CSIF. There is also a driver, */home/bishop/ecs30/getintdriver.c*, that is the same as the one Gradebot uses to link to your function. That driver prints the outputs that Gradebot will use to evaluate your function. Of course, your function should pront *nothing*.

The easiest way to do this assignment is to simply write your function in a file named "getint.c", put the "getintdriver.c" file in the same directory, and type:

```
gcc -ansi -pedantic -Wall -o getint getintdriver.c getint.c
```

Now you have a program with the required interface — assuming your function `getint` works correctly.

1.1 Creating the Gradebot Work Area

First, log into the CSIF.[1] Next, go to where you want the work area directory to be. Then type:

```
git clone metastasis@gradebot.org:user/your_ucd_email/1/4
```

where *your_ucd_email* is your UC Davis email, with the "@ucdavis.edu" deleted. ***Note the "4" at the end of the command; as this problem has ID 4, you need to get the corresponding work area.*** For example, my UC Davis email is "mabishop@ucdavis.edu", so I would type:

```
git clone metastasis@gradebot.org:user/mabishop/1/4
```

to create my Gradebot work area. That area is now in the directory named "4".

1.2 What to Submit to Gradebot

When you want to submit your function for grading, copy it into the directory "4". You should not submit "getintdriver.c". If you do, Gradebot will ignore it, so no harm done. But Gradebot has the same driver, and uses its copy to compile the program. This means, of course, Gradebot will ignore any changes to "getintdriver.c" that you make, even if you submit it.

Once you have "getint.c" in the directory "4", go to that directory and type the following commands:

```
git add getint.c
git commit -m 'change message'
git push origin master
```

The first command informs *git* that you have added a (new or changed) file to the repository.

The second command adds a *change message*, which is a very brief description of what you changed. Remember to put the single quotation marks around the message, or it will not work right (and give you very mysterious error messages! For this step, if you prefer to use the *vim*(1) editor to enter your comment, say instead:

```
git commit -a
```

and the editor will come up.

The third command actually pushes your file over to Gradebot, which will promptly run it. It may take a minute or two, especially if a lot of people are using Gradebot at the time, so look at the "Submissions being graded" and "Submissions to be graded" part of the Gradebot window.

[1] You can do this on your laptop, but the steps may work differently.

2 Submitting the Programs for Grading

Once you have finished the programs and are happy with the results, you have to submit the files to the homework area in SmartSite. You can do this by physically going to the CSIF and uploading them there. Or, you can transfer them to your laptop or home computer and upload them from there. If you do the latter, watch out for character conversion problems. The ASCII printing characters will be transferred correctly, except for the end of line marker; this may be changed to a carriage return, a line feed, or some combination of the two. But if you have non-ASCII characters such as any accented letters, emoticons, or non-Latin letters, those will probably not transfer correctly.

3 Troubleshooting Submitting to Gradebot

If you have trouble getting Gradebot to run your program, but have done so successfully before, then the problem is almost certainly not your key. Try the following steps.

First, be sure you are looking at the current version of the Gradebot page. Gradebot does not send updates to your browser; you have to pull them over. So try refreshing the window in your browser. Check the time when you submitted your last version. If Gradebot didn't run your program, then go on to the next step.

Next, be sure you have set up your work area and you execute the three *git* commands from within that directory. It's very common to forget the first or second command, or to execute the commands in the parent directory, and in these cases the program won't get sent over to Gradebot.

Third, be sure you are executing the *git* commands from the work area. If you see an error like:

```
fatal:  Not a git repository ......
```

this is almost certainly what caused the failure. Find the right working area, change to it, and rerun the command.

If none of these are it, here are a couple of other things to check.

1. You submitted it in the wrong work area. For example, you put "getint.c" in an old working directory, like "2" or "3" (it should be in directory "4'). If you see the following error message in the box on the Messages page, this is almost certainly the reason.

   ```
   make: *** No rule to make target 'getint'.  Stop.
   ```

2. The change comment in the second step is empty. This tells *git* that there have been no changes, and so the third step gives an error, as *git* thinks there is nothing to push over. If you use the `-a` option on this command, you may have put "#" as the first character of the line. Delete that. The "#" at the beginning means the line is a comment and so is ignored.
3. You made no changes to the source code. The program will be pushed over, but Gradebot will compare it to what you submitted before. If you haven't made any changes, Gradebot thinks it has already run the program, and won't do it again. In this case, just find a comment and add a blank to it (or make any other change). Then rerun the three *git* commands again.