All About Programs

This handout describes some general thoughts and techniques for doing programs, as well as what is required, how to submit it, how late submissions are handled, and other administrative matters.

Turning In Programs

All programs are due at 11:55pm on the due date, unless noted otherwise on the assignment. These will be graded and comments returned to you as quickly as possible; we'll try for three class periods, but can't guarantee it.

Please turn in the source code, the *Makefile* (you must include one unless the assignment says otherwise) and any related information (such as manual pages and *README* files). Bundle these into a *tar*(1) file, and submit it. If the file is an uncompressed tar file, choose a name that ends in ".tar". If you compress the file using *gzip*(1) or the `-z` option to *tar*, choose a name that ends in ".tgz". The files should be at the top level of the archive. For example, if your program is in the file *hello.c*, and you have a *makefile* and a *README* file, create the archive as follows (where "X" is the number of the homework):

```
tar cvf hwX.tar hello.c makefile README
```

Be sure that we can recompile the program without errors by typing `make`, unless the assignment says otherwise As we will grade them using systems in the CSIF, please check that your programs run on one of the Linux systems there.

Please turn in your programs electronically using Canvas.

Please do not leave programs for the last minute. The assignments are non-trivial and will require significant design time before you start programming and debugging. When we decide on the due dates, we assume you will spend significant amounts of time on design as well as coding and debugging. If you choose not to do this, you will have difficulty finishing the assignments on time.

Please take the time to design your program carefully. More programming problems arise from improper design than anything else, and the few hours you spend on design will be amply repaid by shorter coding and debugging phases. So please think the design and interfaces through, and—as always—try to find the simplest way to do the assignment (within the limits given in the assignment, of course)!

We do not mind being asked for help; indeed, we welcome it because it helps us know what students are finding difficult or confusing, and sometimes a few words about the problem in class will clarify the assignment immensely. We do mind being asked for help before you have tried to think the problem through. The classic objectionable question (this really happened) occurred on a homework assignment in which the class was given a buggy program. The assignment said the program did not work, and the assignment was to debug it and make it work. That particular class period discussed how to deal with bugs, and gave tips and techniques on how to debug programs. Within 10 minutes of the end of the class during which the assignment was given out, the instructor got this request for help: "The program doesn't run. What do I do now?"

So, before asking for help, please be sure that you have:

1. spent a significant amount of time on the design of your solution;
2. used a debugger if the problem is a programming bug;
3. read all relevant handouts (because your question may be answered there); and
4. tried everything you could think of to solve the problem.

When you come to us, or send us a note, asking for help, please show us whatever you have done to solve the problem, because the first question we will ask you is "What have you tried?" This isn't because we think you're wasting our time. It's because understanding how you have tried to solve the problem will help us figure out exactly what your difficulty is and what we can do to help you. Remember, we will do everything we can to avoid solving the problem for you. When we give you help, our goal is to help you solve the problem yourself.

What We Look For In Programming Exercises

When we grade your homework, we look for simplicity, clarity, elegance, and documentation. Here's a rough weighting of the various factors that go into the grade of each programming assignment:

Correctness	50%
Robustness, including good coding practices	30%
Commenting and ease of reading	10%
Documentation (README, man page, etc.)	10%

We will vary these weights as needed. Please note that correctness is not enough for a perfect score.
If a program does not compile, the most you can get is 20% of the value of the program (10% for commenting and ease of reading, and 10% for documentation). So check your programs before you submit them.

We will drop your lowest homework score from the computation of your final score. See the **General Information** handout for details about how your grade will be computed.

Late Programs

Late homework will be accepted *only* with a doctor's excuse.

Grade Reviews

If you feel that there is an error in grading, please come see me and I'll look over it (and possibly talk with you about it). However, don't dally; any such request must be made within one week of when the grades were made available. After that, we may not change your grade.