Miami University
College of Engineering and Computing
Department of Computer Science and Software Engineering

CSE-278 Systems I Fall 2018

GUIDELINES FOR SUBMITTING HOMEWORKS and LAB EXERCISES

Combined, they represent a significant part of your grade. Read this document carefully and if you have any questions, please ask in class.

Lab exercises are submitted on a weekly basis, time allotted is the next Monday after receiving it. It is likely that you can make good progress or even finish during the Lab itself. They are submitted to the TA using Canvas, which the user is assumed to be familiar. No more details will be given here.

Homeworks are due on the `database.csi.miamioh.edu` server. Thus all the material that constitutes your homework must there, on a specified directory.

You all have accounts on that server. Your account name will be denoted `usrid` in this document.

Rules for Homeworks

1. Projects are due on a *specific date and time*. Requested documents must be on the `submit-hw/userid` directory by the due time.

2. Homeworks are two persons or individual enterprises. In the former case, group is formed. The *leader* is responsible of submitting the project on time in his or her directory. When applies, the two person groups will be assigned by your instructor. New groups will be formed for later homeworks.

3. You must turn in the following:

- Source code `*.cpp, *.c *.h etc.` files.
- Files containing test data (if required)
- Cover sheet.
- A *building script*, or *makefile* with executable permission for all.

These requirements are discussed below.

Naming conventions and building script

The file that contains programs that contain the `main()` method(s) must be named `choiceHN.cpp` , where N is the homework number. *choice* is for your convenience. For example, in the third homework you may want to call the main program `licensesH1.cpp`, the prefix just being a reminder of what the program does.

The building script should be called `build` , and will contain the commands for compilation and generation executable code .

For example, a 1-line building script could be:

```
#!/bin/bash
g++ -o CoolChess -l math cchessH1.cpp figurines.o
```

Change permisions of the script by typing `chmod a+rx build`. Make sure you do this before coping to the grading directory.

Cover Sheet

The cover sheet represents *brief* instructions for a user of your program(s). The cover must reflect the operation of the submitted programs.

Your name(s), date, must be included at the top left. Leader is first, in case you are working in a group.

The cover for the homework must be in a single file `usrmanHN.pdf`, with N being the homework number (typically $1 \leq N \leq 6$. For each program, the cover sheet consists of three parts:

Program description:

A *simple* description of what the (main) programs of the project do. These should not exceed four or five lines. Please do not describe the algorithm; the end-user is normally not interested in this.

Use of the program:

In this section you must answer, most importantly, the following question:
– What does a user (which may have never seen source code) have to do in order to run the program ?
You may assume he has run the `build` script. Other things you may need to address here include answers to:

– What data must he have ready for input?
– What input files are needed?
– How to interpret the output?

If you provide sample input data for your program, specify the name of it here.

Bugs:

If during the testing phase you find out that your program does not run for some input, or produces wrong results, you must report this to the user. In most cases, you should put here "None." Don't let me discover the bugs in your program, as it will cost you points!

Grading of Projects

Each program will be graded according to the following criteria:

1. **Cover** 15%: See above and the example. Notice the use of bold font.
2. **Style** 15%: Proper indentation and comments in the code following the Programming Style Guide at :

   `http://miamioh.edu/cec/academics/departments/cse/academics/programming-style/inde`
3. **Correctness** 70%: The program should be thoroughly tested. Your program must perform to the specifications, with your data or test data provided by the instructor.

Useful Tips

1. Every member of the group must work on *each* of the assigned problems of a project. Then, it must be discussed with the partner, and a decision should be made which "version" must be turned in.

2. Each member is responsible for the whole project. Both members must have the capability of posting a project on his or her submit directory by due date.

3. All homeworks should be available on `database.csi.miamioh.edu` on Monday afternoon. You'll have until midnight next Monday to turn it in. After that, you will not be able to submit, with consequences for your grade.

4. Start working on a project as soon as it is given out. All parts of the project must be submitted together.

5. Make sure you *understand the problem*. Ask the instructor for help with this matter. "Late" help (asking for help on Friday) will not be provided.

6. A program that does not *build* will be given a 0 overall. If it fails to run, the 0 applies to the correctness part only. You might still get some credit for the cover and/or style. Alert: please do not get caught on this! Once you have uploaded into the server, use the script *build* and test it again *there*. Do not assume that if it work in *your* environment (like remote Netbeans IDE) it will work in *my* environment.

7. Proficiency in Systems Programming *is only acquired through practice*. Do all homeworks as well as lab exercises and supplementary problems.

Penalties.
The following deductions will apply to a project, out of 100 points.

Late Projects: 100pts The only things that are going to graded are the contents of your *submit* directory, on the due date.

File without permissions: 15pts first occurrence, 20pts, second ocurrence. **Plagiarism:** 50pts to each party, including the accomplice party. I have an obligation to report you to the Department of CSSE. The penalty is just a suggestion. In a second ocurrence, there may be stiffer penalties.

Missing cover or build script 30pts. Besides that, you will have to demo your program to me.

Example

Suppose that in Homework 2, a problem is is given as follows:

Write a Java program that will compute the product of all positive numbers entered.

You will write a programs for example one that defines the class SimpleIO and , whose source looks more or less like the following:

```
/*  **   Program sumpos **
       Computes product of positive numbers entered

*/

public class  SumposH1 {

    public static void main (String[] argv){

       double pr =1.0;     // partial product
       double x;      // number read
       int n;         // amount of numbers to be entered

       SimpleIO.prompt("Please enter total numbers to be entered")  //prompt
       SimpleIO.readLine(numstr);
       n = Integer.parseInt(numstr);

       for (int i=1; i <= n; ++i){
          System.out.println("Number >");        // prompt
          SimpleIO.readLine(numstr);

          x = Double.parseDouble(numstr);    //conversion
          if (x > 0.0 )
               pr =pr*x;        // accumulating product
       }

       System.out.println("The positive product is " + pr);        //  display
       return ;
     }
}
```

Note the indentation and the good placement of comments. These are the *style* aspects on which your grade is going to be based.

Your *cover sheet* should like like the following. Assuming that the source file is called `sumposH1.java` .

Bob Gruccia 053542
Isabel Ramírez 078211

CSE-278
Fall 18
Homework 1

Program SUMPOS

Program Description: Computes the product of all positive numbers entered. Numbers are entered form the keyborard.

Use of the Program: The user must know in advance how many numbers are going to be entered. This is the first thing that the program prompts for. After this, the program will request the numbers one by one with the prompt:

```
Number>
```

Once it has read all the numbers, the product of the positive ones is printed.

Bugs: If there are no positive numbers entered, rather than issuing some message, it just outputs 1.