Central Washington University
College of the Sciences
Department of Computer Science

CS-302 Adv. Data Structures & File Proc. Fall 2015

GUIDELINES FOR SUBMITTING PROJECTS

A project consists of a programming assignment, which may contain several parts. The projects are a integral part of the course evaluation.

Rules for Projects

1. Projects are due on a *specific date and time*. Requested documents must be on the grading directory by the due time.

2. Projects are two person enterprises. To this end a group is formed. The *leader* is responsible of submitting the project on time in his or her directory. The two person groups might be proposed by your or randomly assigned.

3. You must turn in the following:

– Source code `*.java` files and files containing test data, including, if necessry, other classpaths.

– Cover sheet.

– A building script, with executable permission for all.

These requirement are discussed below.

Naming conventions and building script

File that contains programs that contain `main()` methods must be named `ChoicePN.java` , where N is the project number. *Choice* is for your convenience. For example, in the first project you may want to call the main program `ReadLicP1.java`, 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 of the public classes containing main().

For example, a 2-line building script could be:

```
#!/bin/bash
javac ChessP1.java
javac CountP1.java
```

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.

The cover for the project must be in a single file `usrmanPM.pdf`, with N being the project number (typically $1 \leq N \leq 3$. For each program, the cover sheet consists of three parts:

Program descriptions:
A *simple* description of what the (main) programs of the project do. If the programs are used in sequence, you must indicate that here.

Use of the program:
In this section uou 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 example. Notice the use of bold font.
2. Style 15%: Proper indentation and comments in the code.
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 folder by due date.
3. Start working on a project as soon as it is given out. All parts of the project must be submitted together.
4. Make sure you *understand the problem.* Ask the instructor for help with this matter. "Late" help (less than a week before the project is due) will not be provided.
5. A program that does not compile will be given a 0, overall. It is much better to turn it one or two day late. Once you have uploaded into the server, use the script *build* and test it again there.
6. Proficiency in programming *is only acquired through practice.* Do all projects as well as weekly labs.

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

Late projects: 25pts per day. All parts of project must be submitted at once. Projects will not be accepted after 2 days.

File without permissions: 15pts first occurrence, 20pts, second ocurrence.

Plagiarism: 50pts to each party, including the accomplice party. Second ocurrence, will be reported to the University. See syllabus.

Missing cover or build script 30pts.

Example

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

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

You will write a program, whose source looks more or less like the following:

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

*/

public class  SumposP1 {

    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 `SumposP1.java` . The first named listed is the leader of the group.

Bob Gruccia 053542
Isabel Ramírez 078211

CS-302
Fall 15
Project 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.