1 Introduction

Item-0. NJIT-ID (last 4 digits) and myUCID. Retrieve the last four digits of your NJIT-ID (NOT YOUR SSN). If you don't know your NJIT-ID, login to `my.njit.edu` or read Document 1, Appendix F. For the sake of an example, a homework will be denoted by `hw`, the Programming Project (PrP) by `prp` and the last four digits of an ID by `1234`. In the remainder, we will be using the underscore symbol rather than the dash (minus) symbol, where needed. **The symbol _ is the underscore symbol, not the minus/dash - symbol.** (Java detests dashes and prefers underscores in identifiers, so we use the latter for consistency.)

2 Canvas: Rules and filenames or filetypes.

RULE-0. Accidental submit and resubmit. The last of multiple (hw or PrP) submissions will be graded. The limit however would be kept low (possibly three). Do not send files by email, they will be discarded.

RULE-1. Canvas and Homeworks. You are expected to submit through Canvas. For Homeworks use the Quiz setup of canvas.

RULE-2. Canvas and PrP. You are expected to upload and submit the PrP through Canvas. Only a single .tar or .zip file. Canvas will reject submissions greater than (or equal to about) 5MiB or 5MB.

RULE-3: File-name of a PrP submission.
(i) The first line of every file you submit in the archive must contain your first and last name and the last four digits of your NJIT ID.
(ii) The archive filename would be prp followed by an underscore and the last four of your NJIT ID digits. I would have used for my self a file named `prp_1234.tar`. Every source code file (and for Java, class files) must be named accordingly (eg `Object1_1234.java`, `Code_1234.c`).
(iii) A single text file named for example `prp_1234.txt` within the archive contains a
"I HAVE READ THE COLLABORATION SECTION OF THE SYLLABUS. DOCUMENT 3 and PRP ADHERED."
in capital case as shown (no quotation marks needed). Furthermore, it may contain compilation and/or execution instructions, a bug report and other useful information. We do not read code to evaluate your submission: we just test it.
(iv) Before you submit your archive test it on an OSL/AFS machine. Test it means: dearchive, compile and then run or interpret your code on that machine. Compiling on a GUI environment is different from using the grading environment. For example manipulating text files on Windows or MAC-OSX can cause newline, linefeed problems. In linux you may check for hidden directories with `ls -al`.

RULE-4: Submissions that deviate from RULE-1 through RULE-3 get 0 points.

3 Testing and Grading

4.0 OSL or AFS machine access OFF-CAMPUS. In order to access an OSL machine (eg osl22.njit.edu) or AFS machine (eg afsconnect2.njit.edu) from OFF-CAMPUS, download a VPN client (NJIT supplies one through ist.njit.edu) and install and then activate it as needed by connecting to NJIT. Then use a secure shell client. The site ist.njit.edu contains relevant info or search through Google for example ('NJIT ssh', or 'NJIT VPN'). OSL or AFS machines are Linux machines. They are accessible through the command-line.

Dearchiving and compilation and execution are through the command line (eg unzip, tar, gcc, g++, javac and java).

3.1 Read requirements carefully. The PrP requires **command line processing** and **file-based I/O**. If you are not familiar with them figure this out early in the semester. Submissions that cannot handle command line processing or file-based input/output correctly risk of getting 0 points as ordinary testing will not work.

3.2 Testing and Debugging. Make sure your archived code dearchives (no sub/directories/folders created, hidden or not), compiles and runs on an AFS machine such as `afsconnect1.njit.edu`, `afsconnect2.njit.edu` or `osl22.njit.edu`. This also means you should edit text files, if needed, on AFS using AFS editing tools; avoid remote editing. Do a `gcc -v` or `g++ -v` or `javac -version` or `java -version` to confirm and report in the .txt file the version of compiler used. Versions available on afs are found with a `module avail` and loaded with say a `module load gcc/4.9.2`. If compilation is to proceed in a certain file sequence add a note in the `prp_1234.txt`, as needed. DO NOT DO testing 'remotely' using 'software of the remote platform' to edit files on 'AFS'. If you do not know what a newline becomes in Linux/Unix, Windows, MAC-OSX, be prepared for nasty surprises.

3.3 File types to submit. C or C++ files with `.c` or `.h` or `.cc` or `.cpp` or Java files with `.java` are acceptable. A single text file `prp_1234.txt` must be included per RULE-3. The more info you have in it the less chances you have to get a 0.

3.4 Presence of directory structure is not allowed; file types as in 3.3. Inclusion of binary files (.jar, .class, etc) or other file types than those allowed in 3.3 above will **penalize you 80 points**. Beware of MAC OSX: it has a tendency of generating 0pt PrP submissions because it creates hidden directories. Do not skip step 3.2 above.

3.5 Grading For a HW, grading is more or less straighforward. For PrP, the grader will first decide and create testing instance(s) and then grade your submission based on whether it passes successfully or not those testing instances using the specified interface (eg command-line processing). If your code does not pass any of these testing instances, it will get 0 points, unless there is a detailed bug report that YOU HAVE PROVIDED in `prp_1234.txt`. If no information is provided by you, the grader will NOT read your code.

3.6 Grade and Canvas Grading. The PrP or HW grade will be made available in Canvas. Ignore canvas grade accumulations; canvas has no clue about the course grading scale or scheme. The only deadline is before noon (12-o'clock noon that canvas interprets as 12PM) noon of the day specified in the calendar of Document 1 (Syllabus).] ■