1 Introduction

0. Conventions. The MiniProject is called `mp`. Let the last four digits of your NJIT ID (NOT YOUR SSN) be `WXYZ`. Memorize them! Even if you are a Rutgers student, you still have an NJIT ID. Login to `my.njit.edu` to get your NJIT ID. **The symbol _ we are using is the underscore symbol, not the minus/dash - symbol.** Java detests dashes and prefers underscores in identifiers.

RULE-1: Moodle. Submissions utilize moodle.njit.edu.

RULE-2: Archive File of Submission. Read carefully all of this handout to observe naming and comment conventions and testing guidelines involving the dearchiving of the archive you will submit, compilation and testing on afs machines using testing instances created on those afs machines (manipulating text files on Windows or MAC-OSX can cause newline, linefeed etc problems) and subsequently evaluating the code (eg Sections 2, 3, and 4). The only permissible archive formats are `zip` or `tar`. Thus your submission will contain a `mp_WXYZ.zip` or `mp_WXYZ.tar` attachment.

The archive contains ALL the files of ALL options submitted. **Make sure it DOES NOT include directories/folders or subdirectories hidden or not, or .class/.jar files, or other executable/binary files**. Abide to the naming conventions of Section 2.

A single text file named `mp_WXYZ.txt` within the archive contains either a
"HANDOUT 2 ADHERED; NO BUGS TO REPORT"
in capital case as shown (no quotation marks needed), or compilation and execution instructions, a bug report and other useful information. The first line of this file must be as in Section 3 below.

RULE-3: Submissions that deviate from RULEs 1,2 get 0 points.

2 File and Class names

Every filename, class name must end with the last four digits of your ID. So if we specify in an option that you need to build a class in Java named say `clrs` we in fact mean that your class should be `clrsWXYZ` or `clrs_WXYZ` stored in say `clrsWXYZ.java` or `clrs_WXYZ.java` respectively.

3 The first line(s) of every file you submit

Every file you submit must identify your authoring of the work submitted. Thus if i were you, I would have written something like using either C-style or Java-style comment lines.

```
/* ALEX GERBESSIOTIS   cs435 WXYZ mp  */
or
// ALEX GERBESSIOTIS   cs435 WXYZ mp
```

4 Testing and Grading (of the MiniProject)

4.1. Read requirements carefully. All options require **command line processing** and **file-based input and output**. If you are not familiar with dealing with them figure this out early in the semester and preferably by the midterm. Submissions that cannot handle command line processing or file-based input/output correctly risk of getting 0 points as ordinary testing will not work and the onus would be on you to provide detailed bug information.

4.2. Testing and Debugging. Make sure your 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 `osl11.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` to confirm and report in the .txt file the version of compiler used, or similarly `javac -version` or `java -version`. 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 into the `mp_WXYZ.txt` file.

[Note: 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.]

4.3 Files 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 `mp_WXYZ.txt` must be included. Minimally it includes a first line per Section 3, and the text described in RULE-2 "HANDOUT 2 ADHERED; NO BUGS TO REPORT"
If it is missing, you end up getting 0 points. The more information is available in the .txt file the less chances you have to get a 0.

4.4 Files types or directories NEVER to submit. Inclusion of binary files (.jar, .class, etc) will **penalize you 50 points**. So will the existence of directories/folders or subdirectories/folders hidden or not (MAC-OSX have a tendency to create those).

You are responsible for archiving your files and making sure dearchiving, compilation, and execution takes place properly in any one of the test platforms specified earlier. It is thus imperative to test your archive (and code) in those platforms. If dearchiving, compilation or execution fails, you will not be contacted. Do not expect partial credit without a DETAILED BUG REPORT. Do not complain afterwards.

4.5 No partial credit will be given to submitted code that does not satisfy the previous guidelines. Moreover compliance to the specifications of `mp` is expected including command-line processing and file-based input/output. If we tell you to implement something per class or textbook requirements a deviation will get you a 0. No partial credit if the .txt file does not list bug(s).

4.6 Default Testing Instances and Grading. The grader will first decide and create testing instance(s) on afs and grade your submission based on whether it passes successfully or not these testing instances using the specified interface (eg command-line processing with file-based input/output). 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 `mp_WXYZ.txt`. Any information you provide there, it will help the grader to test your code on some reasonable inputs consistent with the default ones used for everybody else. If no information is provided by you, the grader will NOT read your code.

4.7 Grader and Grade. The grader will email you your MP grade or post it in moodle; you have 5 working days from the day of receipt to contest the score to the grader or the instructor.

4.8 How to get a 0. If a submission is in .rar, or uses HashMaps or sorting of any type, or performs linear-time operations where a constant-time solution is possible (eg using Vector operations inefficiently) will get you 0 pts: **Do not complain then.**

Neither the instructor nor the grader will answer questions of the type: `How do i debug a program? What is a segmentation violation?` For such questions contact your CS100, CS113, CS114, CS288 instructor(s)!