Name:____

7. This problem asks you to design a Makefile and version-control scheme for automatically generating documentation for Java code.

Scenario:

- Assume `a.java` defines one class `A`, and `b.java` defines one class `B`.
- The `javadoc` program takes a Java file (e.g., `a.java`) that defines a class and makes an HTML file that describes the class (e.g., `a.html`).
- You need to add a license agreement to the top of every HTML file that `javadoc` produces. The contents of the license are in a file `license`. You have written a shell-script `add-license` that takes an HTML file and changes it so it includes the contents of `license`.

(a) **8pts** Write a `Makefile` with targets for making `a.html` and `b.html`. The generated files should include the license. They should be remade whenever and only whenever a file that could affect their contents has changed.

(b) **4pts** Which of the files mentioned in this problem would you put in a version-control system? Briefly justify your inclusion or exclusion of each file.

Solution:

(a)
```
a.html: a.java license add-license
        javadoc a.java
        add-license a.html

b.html: b.java license add-license
        javadoc b.java
        add-license b.html
```

(b) `a.html` and `b.html` should *not* go in the repository because they are automatically generated. All the other files should: The Java and license files are inputs to make the HTML files. add-license is a program written for this task; its contents affects the result. Also, the `Makefile` should go in the repository so other developers can use it. (No points deducted for not discussing `javadoc`, which should not go in the repository because it is an executable and is a tool used (not developed) by this project.)

Name:____

3. (**15** points) Write a `Makefile` for this scenario:

   - An application `myprog` is written in C, with all the code in `myprog.c`.
   - You wrote two test-inputs, in files `input1` and `input2`.
   - You want to run `myprog` *with profiling* on each test-input and then use `gprof`, saving the result to file `prof1` (for input `input1`) or `prof2` (for input `prof2`).
   - You have a bash script `compare` that takes as arguments two files created by `gprof` and produces an interesting summary. You want a phony `run` target that runs `compare` on `prof1` and `prof2`.

   Your `Makefile` should re-compile or re-run programs only as necessary (except the `run` target should always execute `compare`), but it should never use out-dated programs.

   Hints: You should have 4 targets. Some will need multiple commands. Some will need multiple sources.

   Solution:

```
myprog: myprog.c
        gcc -o myprog myprog.c -pg

prof1: myprog input1
        ./myprog input1
        gprof myprog > prof1

prof2: myprog input2
        ./myprog input2
        gprof myprog > prof2

run: compare prof1 prof2
        compare prof1 prof2
```