CS99, Fall 2001: Homework 1

Due Tuesday 9/18 in lab

1. Objectives

Completing all tasks in this assignment will help you learn about the process of programming before you do actual writing of code. In this assignment, we expect you to do the following:

- Follow all instructions for doing this assignment.
- Pick a project name, which eventually becomes a program name.
- Write a clear algorithm without too much or too little information. We know that this might be your first attempt, so we will ask for this objective on more assignments.
- Demonstrate that you understand the process involved with solving a problem before writing a computer program.

First skim, and then, carefully read the *entire* assignment before starting any tasks!

2. Algorithms and Problem-Solving Process

Topics: the problem-solving process, algorithms, stepwise refinement

Problem: Suppose that you work for a shipping company that is contracted to develop a state-of-the-art simulation to help improve its operation efficiency.

Background: The company needs to ship boxes to consumers but is too cheap to hire more than one worker. At the start of one cycle, a random number of boxes (between 1 and 3, inclusive) spews forth from a chute into a bin with no boxes initially inside. The lonely worker must take boxes out of the bin and place them in a truck for shipping, which ends the current cycle. Because of sporadic back pains, the worker will take a random number of boxes (between 1 and 3, inclusive) out of the bin. Although the worker starts completely refreshed, all this work tires the worker. The worker's ability to carry boxes starts at 100%. But, after every four cycles, which means four trips between the bin and truck, the worker's efficiency drops by 25% of the previous value. As a result, the worker's ability to carry boxes decreases eventually to the point when the worker will make a trip, but pick up zero boxes. Unfortunately, the bin may hold a maximum of 7 boxes. The worker keeps attempting to extract boxes as long as possible to prevent the bin from exceeding its capacity.The operation shuts down when the bin overflows, which happens because the worker stops carrying enough boxes.

Tasks: Your boss has dumped this project on you because you are the only one in the company who knows how to program, since you took CS99. You should be in good shape, especially because you will be performing the steps of basic software engineering, as described in the sections below. You will be asked to do different tasks in each section.

2.1 Naming

Pick a project ***name***, which will be the name of the class that contains the `main` method, which is sometimes called the ***Main Class*** or ***target***. You must first arrange a time to meet with your "development group," which, of course, is either just yourself for this assignment. Every program must have a cool-sounding name! Put this name on the following steps.

2.2 Brainstorming

Programming resembles writing a paper. What is the first step in writing? ***Brainstorming***. For programming, brainstorming means writing down explicit and implicit specifications. Examples of explicit specifications include the following:

- The computer picks a random number between 1 and 3, inclusive.
- The simulation ends when the bin contains eight or more boxes.

Often, information is implicit, unclear, or under-specified. You might ask some of the following questions:

- Is the computer picking an integer or float?
- Is there a limit to the number of cycles that the simulation can perform?
- Does the program need to keep track of the number of cycles?

To help you develop this program, write down as many specifications as you can think of for this simulation, including the ones supplied to you, above. You might have to make some of your own decisions or request further clarification from your client (the person/company/source asking you to develop your code). Include this write-up in your project as the first part of this solution. Do not forget to include the name of your program at the top.

2.3 Research

After brainstorming, review your specifications and thoughts and look for *things* and *behaviors*. Sometimes I like to refer to this search as a game called "Spot the nouns and verbs!" (And thus, I deem this procedure as ***research***.) These nouns/things and behaviors/verbs become important parts of your program. Until we start object-oriented programming, you will likely represent each *thing* with a variable and each *behavior* with an operation or method call. You may wish to do actual research where you look up prepackaged MATLAB functions for generating random numbers and any other part of the language that might help you. "Research" the problem, brainstorm, and record a list of *things* and *behaviors* that you identified.

2.4 Outlining

The next stage of writing involves organizing your thoughts from brainstorming and research into an outline. For programming, these outlines are essentially ***algorithms***. So, take the specifications, and try to figure out how you, the human being, would model this simulation. Before you can automate the process, you must figure out how the simulation works, preferably on paper. You may even wish to simplify the problem first.

To write an algorithm, organize your solution procedure and write each step as an instruction. If you have trouble writing the instruction, try to imagine that you are writing a recipe or manual for another person to follow. Each instruction should start with a verb (start, stop, choose, repeat, pick, output, return, find, etc.) or condition (if, else, otherwise, unless, until, while, etc.). When you pick these words, you are writing in ***pseudocode***, which is a language that resembles a programming language. You may even wish to indent the pseudocode in blocks as if you were writing in actual code. Ideally, given well-written pseudocode, you could quickly convert any algorithm to actual code in almost any language.

If writing on paper is too difficult, you could try a trick that DIS uses to appease the desire to program immediately. Create a new file where you would write the code, but do not actually starting coding yet. Instead, type the brainstorming notes, specifications, research, and algorithm as a large block of comments. Gradually, flesh out more and more of the algorithm until you feel ready to program. Eventually, you could feasibly (and quickly) write the necessary code a bit at a time, test that code, and move to the next portion. In the end, you will not have produced working code, but comments as well!

However, for this portion of the project, do not actually write the code! Instead, clearly write out your algorithm. In the next project, you will use the work in this project as we complete our demonstration of the entire development process. You will be drafting, writing, polishing, testing, and submitting your code.

3. Submitting Your Work

Submit all work that you performed in this assignment. Follow the submission guidelines, which are stated in **Assignments→Homework Format** and **Assignments→Submitting Assignments**.