Project 1A Specification

Goals

The goal of this project is to develop—through experience—your ability to:

1. Make technical choices.
2. Bring multiple ideas together to solve a problem.
3. See how one problem can serve as a precedent for the solution of another.
4. Deal with problems that have less specification than that typical of a problem set.
5. Organize and perform work with a partner, if you choose to work with a partner.
6. Write about your work.

The Resource-Allocation Problem

Resource allocation is an often-difficult, real-world problem. One instance of this problem is the assignment of part-time workers to time on a job. In its simplest form, you can imagine a set of workers, each of whom wants a shift to work, and a set of shifts, each of which can be assigned to a worker.

You could think of this as a search problem, with depth *w* and branching factor *s*. You could think of solving the search problem by doing a British Museum search; that is, you could generate all possible paths and then evaluate each. Alas, even with just 10 workers and 15 shifts to schedule, a British-Museum enumeration of assignments unthinkable, which you should show yourself, if in doubt, on the back of an envelope.

You could also think of this as a constraint satisfaction problem. In this case, either workers are the variables, with shifts the values in their domain, or shifts are the variables and workers assigned to them are the values.

The Problem

Consider MIT dormitory desk workers...

For those who aren't familiar with the system:

Each dorm has a Desk, which is by the front door to the dorm, and the desk worker is responsible for tasks like screening who gets through the door, answering the phone, and as a host of other small duties. The Desk needs to be manned throughout most of the week, and is generally manned by students who want or need part-time work.

There is a weekly schedule generated by the Desk Captain (the student given direct authority over the Desk), which tells the desk workers who is working and when. If a student can't make his regularly scheduled shift, it is his responsibility to find somebody else to cover for him.

As it happens, making up the schedule for desk workers by hand is a long and tedious task, of just the sort that 6.034-equipped computers can solve. For years now, the Random Hall Desk Captains have been swearing that they will write such a program so that the Desk Captains who follow won't have to do hand assignment. That time has finally arrived: your task is to design the program which assigns students to work the Random Hall Desk.

This is a real problem, and, if your project works well, we'll send it to Random Hall and they may decide to use it.

What you start with

We provide you with a description of how desk workers are scheduled and a case study example of preference sheets and resultant hand-created schedule We also provide you with a possible search-based starting point in the form of a program cum data set and rule set with the following characteristics:

1. The search done is depth-first search.
2. Node expansion is done by a rule-based system.
3. There is sample database of a few assertions and one sample rule.
4. The assumption is that no student wants to work more than one hour.

Your job

You will need to do something intelligent to assign desk workers. In particular, you are to do the following:

1. Develop a set of assertions from the student and scheduling-rule descriptions. You must decide what to make use of in the descriptions and what to ignore: you may decide to capture some details of description only approximately or not at all.
2. Develop a set of rules using commonsense and the student and scheduling-rule descriptions.
3. Choose and implement a search or constraint strategy.
4. Choose and implement a mechanism for determining the quality of a particular desk worker assignment.

You may be able to find a solution for the sample data by hand. Keep in mind that the goal is not only to find a solution for the sample data, but also to write a program that will work in future semesters with, for example, other, more, or fewer preferences and requirements.

Also, note that if your program is successful, it may be used to do part-time worker assignment for larger groups of workers and schedules than a single desk worker at a small dormitory.

Check points

The following are default checkpoint days; you may negotiate small movements with your TA. We note that a quiz occurs near the first date.

Oct 11	You are to have completed your knowledge engineering work, which you will submit to your TA.
Oct 21	You are to have implemented an interim version of a running, documented program and produced one or more sample runs, which you will show to your TA. Your interim implementation is to handle requirements, but not necessarily preferences.
Oct 28	You are to have completed a report along with a final version of your running, documented program and produced one or more sample runs.

Completed your knowlege engineering work means deciding on a representation. If you use rules, you need to show representative rules in Scheme or pseudo-scheme (or Java or other language by agreement). If you use constraints, you similarly need to show representative constraints in Scheme or pseudo-scheme (or other language). You are not bound by what you submit on Oct 11; the first check point is in place to ensure you don't try to do all you thinking the night before the project is due.

Report Length

The right length for a paper is always the shortest length that covers what you want to say clearly. As a rough guide, we do not want you to write more than **five pages**, exclusive of illustrations, code, run printout, and the like. We would be surprised if you can say what needs to be said in much less.

Important Notes

1. Your program is to distinguish between **requirements** and **preferences**: requirements cannot be violated; preferences can be, but the quality of a set of worker assignments declines in proportion to the number of violated preferences.

2. Your program is to do the best it can with more workers than shifts, just the right number, or not enough workers for the needed shifts.
3. You may wish to solve the problem in two steps, first assuming that each student only wants to work one shift; then, generalize to be able to assign multiple shifts to one student.