Leon Zhang  
llz210000

Project 2 Write-up

## Approach

My approach to this project was to break the control flow of the program into steps that I could tackle one by one. This way, I could get a good feel for how to do checks and add employees to workstations.

## Organization

I organized the project into 3 main sections: the plan/3, adding the workstations to each shift, and adding employees to each workstation. All these would interact with each other, so planning out what each predicate needed was crucial. The plan/3 predicate called adding workstations for each shift type and each workstation called a predicate to add employees based on the min, max. Additionally, I other helper predicates that checked for things such as if an employee could be added to a certain workstation.

## Reverse Predicates

Cuts are not reversible in Prolog because cuts make the program choose a path that it can’t backtrack from.   
Most arithmetic predicates cannot be done in reverse because of the way they are written, ex: “X = Y + Y” called by add(x,y) written as add(x,3) will give 6 but add(3,y) fails to run because y is not instantiated as any value.

## Problems

A big problem was grasping how to manipulate lists in different was and use them in parameter calls. While the [Head | Tail] convention was not unfamiliar, it was still a struggle to work with lists without the usual accessibility you get when using an OOP language. Additionally, the overall syntax was still confusing at times and took a while to adjust to. Conceptually, visualizing the problem was a lot harder than expecting because of all the different outcomes that were possible and translating it to code in an unfamiliar language was a difficult task.

## Solutions

Like the project before, working through the project and experimenting with different syntax and code helped a lot to gain familiarity with the language. For conceptualizing, I think I was able to figure out a general good program flow for how I wanted the logic to work, but was not able to implement it as well.

## What I Learned

I learned a lot about logical programming and the things I liked and disliked about it. The main thing this project made me learn was how to work well with recursion because many predicates I used were built on recursive calls and I learned how to better manipulate lists and other data through those calls. For things I liked, I think finding solutions to things in Prolog was very satisfying because of how intuitive and relatively simple the syntax is. For things I disliked, many were the struggles of working with a new language and especially working with logical programming. Without easy access to global variables and knowledge of built-in functions, I felt like a lot of things I wanted to do I was not able to figure out how to and it made progress really slow.

## Incomplete Features

I did not implement cuts in my code. I did not really understand how cuts worked and didn’t try to implement them until late. When trying to implement, I would always get empty lists as output and had trouble bug fixing, so for the sake of time, I stopped trying.   
My program output is also incorrect for many things. I ran into the problem of not being able to keep track of the employees who were already assigned to a workstation and ended up having the list of employees repeat every shift. This in turn made all the testing cases not work properly either. This is the main problem I faced and spent most my time trying to fix. My latest attempt to fix it in my submission code was extracting just the names from each shift and passing that onto the next shift to check if an employee was in the list so they would not be added again, but the code does not work properly and I couldn’t figure out why. I am unsure if there is just a big logical error in my code or if it is something small that I have just missed this whole time. Another solution I tried was to delete the employees that I already added to a work station but I was not able to get that to work properly either.