CSCI 4430/6430 Programming Languages
Fall 2019
Programming Assignment 3
Deadline: 7:00pm EST December 02, 2019

This assignment is to be done either individually or in pairs. Do not show your code to any other group and do not look at any other groups code. Do not put your code in a public directory or otherwise make it public. However, you may get help from the mentors, TAs, or the instructor. You are encouraged to use the Submitty Discussion Forum to post problems so that other students can also answer/see the answers.

This assignment will strengthen your understanding of logical and constraint based programming through application in either Prolog or Oz. You will be tasked with solving systems that have constraints specified in a subset of natural language.

Please be honest with your submitted code. Any solution that simply prints out the correct answer without performing calculations that lead to the correct answer will result in a zero for the entire assignment.

Part 1: Curious Set of Integers

This section serves as a warm up to get you used to working with Prolog/Oz under a constraint based programming model. You will be presented with a simple puzzle (representing a constraint programming problem) that you must solve.

Puzzle Description

The original version of this puzzle was posed by Martin Gardner in February 1967.

The integers 1, 3, 8, and 120 form a set with a remarkable property: the product of any two integers is one less than a perfect square. Find another set containing four unique numbers between 0 and 10000 that also has this property and it contains no numbers from the original set of {1, 3, 8, 120}.

Code Requirements

Your code should print the results to the standard output as comma separated values.

Example Output

```
$ 1,3,8,120
```

Part 2: Parsing Constraints and Building Interesting Sets

One of the advantages of logical programming is that parsing languages comes naturally from the computational model. In this section you will be tasked with creating a parser for a specified grammar and using the resulting parse tree to solve a simple constraint programming problem.

The grammar that you will be using accepts sentences that define the constraint programming problem of finding a set of integers with specific properties, called **interesting** sets. Some examples of sentences that the grammar accepts are: `Find a set of 2 odd integers that sum to 16` and `Find a set of 2 even and 3 odd integers that multiply to 120`. Note that this grammar accepts sentences that correspond to a constraint programming problem with no solutions: `Find a set of 2 even integers that sum to 3`. To help make your search space smaller you can assume that each element in the set has a value between 0 and 128 inclusive. Note that the values in a set must be unique and their order does not matter.

Interesting Sets Grammar

- **S** → Find a set of **I** that **Op** to **Number**
- **I** → **Even** integers | **Odd** integers | **Both** integers
- **Even** → **Number** even
- **Odd** → **Number** odd
- **Both** → **Even** and **Odd**
- **Op** → sum | multiply
- **Number** → 1 | 2 | 3 | ...

Code Requirements

Your code should accept as input a string that defines the constraints of a particular interesting set of integers. There are three possible types of output for your program depending on if the string is invalid in the grammar, there is no solution to the problem, or at least one solution exists:

- Invalid input: `$ Invalid String`
- No solution to the problem: `$ No Solution`
- At least one solution exists: Comma separated list of values (shown in section below)

Example Input/Output

Input:
`Find a set of 3 odd integers that sum to 15`
Output:
`$ 3,5,7`

General Code Requirements

Each part of this assignment should have its own source code file named ***part_x.oz*** or ***part_x.pl*** (x = 1, 2). Output for each file should go to standard output and more detailed specifications may be found in the corresponding sections above.

Sample code has been provided on the **PA3 website**. This starter code takes as input a single string and then prints that string to standard output. In order to run the starting code follow the same instructions provided below in the Running Code sections.

Setting Up Oz

The most up to date version of Oz/Mozart (2.0) does not allow for constraint programming. Consequently, you are required to use Mozart 1.4.0 which only runs on 32-bit operating systems. Since most modern systems are 64-bit, we have created a virtual machine in the **VirtualBox 6.0** provided **student.ova** file. This will provide you with an environment that already has Oz/Mozart 1.4.0 fully installed and ready to go. To log into the VM the first time the password is **student** and will remain that way unless you change it.

Running Oz Code

Your code will be run in the following way on Submitty:

- `$ ozc -c part_1.oz`
- `$ ozengine part_1.ozf > part_1.txt`
- `$ ozc -c part_2.oz`
- `$ ozengine part_2.ozf "Example Input String" > part_2.txt`

Setting Up Prolog

If you plan to use Prolog for this assignment you are required to use SWI-Prolog which can be found **here**.

Running Prolog Code

- `$ swipl -q -f part_1.pl -t main > part_1.txt`
- `$ swipl -q -f part_2.pl -t main "Example Input String" > part_2.txt`

Grading

Submissions will be graded according to the following rubric:

Part 1	20%
Part 2	70%
Code Clarity/ Comments	10 %