EAST TENNESSEE STATE UNIVERSITY

Lab 1: Hello, AI Agent

Overview

You and your colleagues need to have a meeting. However, because you're all busy people, your schedules are difficult to coordinate. In addition, you only have access to one conference room, which has a list of other meetings. Your job is to create an intelligent agent that takes everyone's schedules and determines a list of times when you can all meet in the conference room. To do this, you must lay out each colleague's time schedules, and the room schedule, then list only the times that everyone is available.

Generate and Test

The generate-and-test algorithm is one of the simplest methods of constructing an artificial intelligence agent. The algorithm is basically a bunch of loops, which is obviously not very efficient for larger problems. For simpler problems, generate and test works. A description of the algorithm and pseudocode follows.

```
# Generate And Test Algorithm
# --------------------------
# == Generate possible solutions and test each solution to see if it meets
#    the goals set forth.
# Inputs:
#  - Any number of Lists
#  - At least one Goal Function (that returns a Boolean)
#  - Any number of Constraint functions (that each returns a Boolean)
# Outputs:
#  - Optional
# Effect:
#  - For all possible combinations of each list,
#    o Check to see if it's the goal combination, and
#    o Check to see that it satisfies the constraints

def generateAndTest(list1, list2, …, listN,
          goal1(), goal2(), …, goalN(),
          cons1(), cons2(), …, consN()):
   for a in list1:
      for b in list2:
         …
         for n in listN:
            if (goal1(a,b,…,n) and … goalN(a,b,…,n) and
                cons1(a,b,…,n) and … consN(a,b,…,n)):
               # Solution Located
```

Step 1: Modeling the Environment

The Schedules

The following table represents the times each person, and the conference room, is **available** for each day. The indexes provided indicate the indexes you should use in your multi-dimensional list that represents the schedule.

	Monday [0]	Tuesday [1]	Wednesday [2]	Thursday [3]	Friday [4]
Conference Room [0]	8, 9, 13, 15	9, 11, 16	8, 9, 10, 12	10, 12, 13, 15, 16	8, 9, 10, 11, 12, 13, 14, 15, 16
Yourself [1]	8, 12, 13, 15, 16	8, 9, 11, 13, 14	8, 12, 13, 15, 16	8, 9, 11, 13, 14	8, 9, 11, 13, 14, 15
Anna [2]	9, 10, 11, 13, 15	9, 10, 11	8, 9, 10, 11, 13, 14, 15	9, 10, 13	8, 9, 11, 12, 15
Bob [3]	10, 11, 13, 15	8, 9, 10, 11	10, 11, 13, 15	8, 9, 13, 15	8, 9, 11, 13
Carrie [4]	8, 9, 10, 13, 14, 15	11, 12, 13, 14, 15	8, 9, 10, 13, 14	12, 13, 14, 15	8, 9, 10, 11, 13

Converting the Schedules to Python Lists

Create a Python file named **lab1.py**. Using a multidimensional list, create one list for each person and one for the conference room. Each list should contain five lists—one list for each day—each containing the available times for the person for that day. To make this uniform, use the indexes indicated in the table. For example, `schedule[1][2]` is a list of available times for yourself on Wednesday.

Step 2: Define the Generate and Test Function

Create a Python function named `generateAndTest` that accepts five lists as parameters. Each list is a person's availability for a particular day of the week. Use the pseudocode above to define the method. Note that, in this case, we only have ONE goal, and NO extra constraints.

To represent the `goal`, create a `goal` method that accepts five numbers (representing times) as parameters. The goal test should return TRUE if all five numbers are equal. You will integrate this into your `generateAndTest` function.

Step 3: Producing Output

Run the `generateAndTest` function for each day. Your output should appear as follows. (This output is not reflective of the solution). Note that a day with no meeting times should simply be left blank.

```
MONDAY:
- POTENTIAL MEETING at x:00

TUESDAY:
- POTENTIAL MEETING at x:00
- POTENTIAL MEETING at y:00

WEDNESDAY:
- POTENTIAL MEETING at x:00

THURSDAY:
```

```
- POTENTIAL MEETING at x:00

FRIDAY:
```

Lab Report

Create a document called **Lab1.docx**. Answer the following questions.

1. Agent Type (10 Points)

In the lecture, we discussed the following types of agents: Simple Reflex Agent, Reflex Agents with State, Goal-Based Agents, and Utility-Based Agents. What type of agent is the one you created for this lab? Why? Would any other type of agent also apply? Why?

2. PEAS Description (15 Points)

One way to describe the task environment for an agent is the PEAS description. Use the following table to complete the PEAS description for this assignment's agent. See the book—Section 2.3.2, or the slides.

Performance Measure	
Environment	
Actuators	
Sensors	

3. Book Problems (15 Points)

Complete the following problems from the book, page 61-62:

2.3, 2.4, 2.5

Comment on the efficiency of the generate-and-test algorithm. Mention the Big-O of the program you wrote, if possible.

Submission

Submit both files, **lab1.py** and **lab1.docx** to the dropbox on D2L by **Sunday, January 20 at 11:59pm**. Assuming you provide a reasonable effort for each problem, and a reasonable answer, you will receive full credit for the question. The rubric is on the following page

Rubric

	Points Possible	Points Gained	Comments
Code			
Generate-and-Test Correctly Defined	15		
Correctly Uses multidimensional lists to represent times	15		
Goal method	15		
Results are correct	15		
Lab Report			
1. Agent Type	10		
2. PEAS	15		
3. Book (2.3)	5		
Book (2.4)	5		
Book (2.5)	5		