Programming Languages Homework: Programming Assignments
Spring 2017
Rowan University

You may work together on these programming assignments, in teams of up to 3 people. If you work in a team, e-mail only one copy of the program and submit only one printout with all the team members' names on them.

This semester, you will solve the same programming problem three times: once in Scheme, once in Ada, and once in Prolog.

General problem description

You will write a program which checks a work assignment against a collection of constraints to see whether the work assignment is acceptable.

In the IT Department of the Magrathean Construction Company, there are four work shifts during the day: 9:00-11:00, 11:00-1:00, 1:00-3:00, and 3:00-5:00. During each shift, four workers are needed: Two to answer customers' questions on the phone, one to repair computers, and one to fix network problems. No worker can work more than one job or shift during a day. Thus, it takes 16 different workers to fill all the jobs during a day.

The various employees of the company have different skills: some are good at answering phones, some are good at repairing computers, some are good at fixing network problems, and some are good at more than one of these tasks. An employee should only be assigned to a job if he/she has that job skill.

The input to your program will be a proposed job assignment (the phone, computer, and network workers for each of the four shifts), followed by a list of workers and their skills. The workers and their skills will be specified by a worker's name, then three numbers (1 or 0) indicating whether the person is good at answering phones, repairing computers, and fixing the network, in that order. 1 indicates that the employee has the skill, and 0 indicates that the employee does not have the skill.

The output will be a message indicating whether the proposed job assignment is acceptable.

Sample calculations

Example 1:

Input:

```
Aziz Collins Vieira Liu
Davis Ericson Kelly Singh
Fitzgerald Gutierrez Martinez Tortorella
Hughes Jones Patel Zimmerman
20
Aziz 1 0 0
Blumenthal 0 1 1
Collins 1 1 1
Davis 1 0 0
Ericson 1 0 0
Fitzgerald 1 0 1
Gutierrez 1 0 1
Hughes 1 0 0
Jones 1 1 1
Kelly 0 1 1
Liu 1 0 1
Martinez 0 1 1
Patel 1 1 1
Qureshi 1 1 0
Rodriguez 1 0 0
```

```
Singh 0 1 1
Tortorella 0 0 1
Vieira 0 1 0
Young 0 1 0
Zimmerman 1 1 1
```

Output:

```
Acceptable
```

Rationale:

The proposed work assignment says that

> In shift 1, Aziz and Collins will answer phones, Vieira will fix computers, and Liu will fix the network.
> In shift 2, Davis and Ericson will answer phones, Kelly will fix computers, and Singh will fix the network.
> In shift 3, Fitzgerald and Gutierrez will answer phones, Martinez will fix computers, and Tortorella will fix the network.
> In shift 4, Hughes and Jones will answer phones, Patel will fix computers, and Zimmerman will fix the network.
>
> The number 20 means that there are 20 employees in the list that follows.
>
> Aziz is good at answering phones, but cannot fix computers or the network.
> Blumenthal is not good at answering phones, but can fix computers and the network.
> Collins is good at all the jobs.
> Tortorella, the 16th employee in the list, is not good at answering phones or fixing computers, but can fix the network.
> Vieira, the 18th employee in the list, is not good at answering phones or fixing the network, but can fix computers.
> Zimmerman, the 20th employee in the list, is good at all the jobs.
>
> Everyone has been assigned to jobs that they can perform, and no one has two jobs, so the assignment is acceptable.

Example 2:

Input:

```
Young Collins Vieira Liu
Davis Ericson Kelly Singh
Fitzgerald Gutierrez Martinez Tortorella
Hughes Jones Patel Zimmerman
20
Aziz 1 0 0
Blumenthal 0 1 1
Collins 1 1 1
Davis 1 0 0
Ericson 1 0 0
Fitzgerald 1 0 1
Gutierrez 1 0 1
Hughes 1 0 0
Jones 1 1 1
Kelly 0 1 1
Liu 1 0 1
Martinez 0 1 1
Patel 1 1 1
```

```
Qureshi 1 1 0
Rodriguez 1 0 0
Singh 0 1 1
Tortorella 0 0 1
Vieira 0 1 0
Young 0 1 0
Zimmerman 1 1 1
```

Output:

```
Not Acceptable
```

Rationale:

This is the same input as in example 1, except that Young has been assigned to answer phones in shift 1 instead of Aziz. The assignment is unacceptable because Young is not qualified to answer phones.

Example 3:

Input:

```
Aziz Collins Vieira Liu
Davis Aziz Kelly Singh
Fitzgerald Gutierrez Martinez Tortorella
Hughes Jones Patel Zimmerman
20
Aziz 1 0 0
Blumenthal 0 1 1
Collins 1 1 1
Davis 1 0 0
Ericson 1 0 0
Fitzgerald 1 0 1
Gutierrez 1 0 1
Hughes 1 0 0
Jones 1 1 1
Kelly 0 1 1
Liu 1 0 1
Martinez 0 1 1
Patel 1 1 1
Qureshi 1 1 0
Rodriguez 1 0 0
Singh 0 1 1
Tortorella 0 0 1
Vieira 0 1 0
Young 0 1 0
Zimmerman 1 1 1
```

Output:

```
Not Acceptable
```

Rationale:

This is the same input as in example 1, except that Aziz has been assigned to answer phones in shift 2 instead of Ericson. The assignment is unacceptable because Aziz has been assigned to two jobs in a single day.

Example 4:

Input:

```
Aziz Collins Vieira Liu
Davis Ericson Kelly Singh
```

```
Gutierrez Gutierrez Martinez Tortorella
Hughes Jones Patel Zimmerman
20
Aziz 1 0 0
Blumenthal 0 1 1
Collins 1 1 1
Davis 1 0 0
Ericson 1 0 0
Fitzgerald 1 0 1
Gutierrez 1 0 1
Hughes 1 0 0
Jones 1 1 1
Kelly 0 1 1
Liu 1 0 1
Martinez 0 1 1
Patel 1 1 1
Qureshi 1 1 0
Rodriguez 1 0 0
Singh 0 1 1
Tortorella 0 0 1
Vieira 0 1 0
Young 0 1 0
Zimmerman 1 1 1
```

Output:

```
Not Acceptable
```

Rationale:

This is the same input as in example 1, except that Gutierrez has been assigned to answer phones in shift 2 instead of Fitzgerald. The assignment is unacceptable because Gutierrez has been assigned to two jobs in a single shift. The phones must be staffed by two different people in each shift.

Example 5:

Input:

```
Park Dunbar Khan Lloyd
Bashir Trivedi Ramos Franklin
Schmidt Nakamura Andrews Chen
Evans Goldberg Macmillan Ortiz
17
Andrews 1 1 1
Bashir 1 1 1
Chen 1 1 1
Dunbar 1 1 1
Evans 1 1 1
Franklin 1 1 1
Goldberg 1 1 1
Jefferson 1 1 1
Khan 1 1 1
Lloyd 1 1 1
Macmillan 1 1 1
Nakamura 1 1 1
Ortiz 1 1 1
Park 1 1 1
Ramos 1 1 1
Schmidt 1 1 1
Trivedi 1 1 1
```

Output:

```
Acceptable
```

Rationale:

This example is easier to check, because everyone is able to do every job. There are 17 employees this time, and any selection of them that does not contain duplicates will be acceptable.

A note on I/O format

The exact format of the input and output will be slightly different for the three different languages, to make the input and output as easy as possible in each different language. The three input/output specifications for the three different languages are given below.

In all versions of the problem, assume that the input will be correctly formatted; your program does not need to check for formatting errors.

---

1. Scheme problem

Write a Scheme program to solve the problem described above. Define a function *(jobs_ok? Assignment Employees)* which takes two parameters, *Assignment* and *Employees*. *Assignment* is a list of four sub-lists, one for each shift; each sub-list contains the work assignment for that shift: 2 names for phones, followed by one for computer repair, followed by one for network repair. *Employees* is a list of sub-lists; each sub-list contains an employee's name, followed by three integers (1 or 0) indicating whether the employee is qualified for phones, computers, and networks, respectively. The function should return #T if the job assignment is acceptable and return #F if the job assignment is not acceptable

You may assume that the *Assignment* and *Employees* lists will be in the correct format when the function is called; you do not have to error-check for a non-list or an incorrectly formed list. In your program, you may write and call any additional functions that are helpful in the computation.

Examples:

```
(jobs_ok?
   '(
        (Park Dunbar Khan Lloyd)
        (Bashir Trivedi Ramos Franklin)
        (Schmidt Nakamura Andrews Chen)
        (Evans Goldberg Macmillan Ortiz))
   '(
        (Andrews 1 1 1)
        (Bashir 1 1 1)
        (Chen 1 1 1)
        (Dunbar 1 1 1)
        (Evans 1 1 1)
        (Franklin 1 1 1)
        (Goldberg 1 1 1)
        (Jefferson 1 1 1)
        (Khan 1 1 1)
        (Lloyd 1 1 1)
        (Macmillan 1 1 1)
        (Nakamura 1 1 1)
        (Ortiz 1 1 1)
        (Park 1 1 1)
        (Ramos 1 1 1)
        (Schmidt 1 1 1)
        (Trivedi 1 1 1)))
```

should return #T, and

```
(jobs_ok?
   '(
        (Young Collins Vieira Liu)
        (Davis Ericson Kelly Singh)
        (Fitzgerald Gutierrez Martinez Tortorella)
        (Hughes Jones Patel Zimmerman))
   '(
        (Aziz 1 0 0)
        (Blumenthal 0 1 1)
        (Collins 1 1 1)
        (Davis 1 0 0)
        (Ericson 1 0 0)
        (Fitzgerald 1 0 1)
        (Gutierrez 1 0 1)
        (Hughes 1 0 0)
        (Jones 1 1 1)
        (Kelly 0 1 1)
        (Liu 1 0 1)
        (Martinez 0 1 1)
        (Patel 1 1 1)
        (Qureshi 1 1 0)
        (Rodriguez 1 0 0)
        (Singh 0 1 1)
        (Tortorella 0 0 1)
        (Vieira 0 1 0)
        (Young 0 1 0)
        (Zimmerman 1 1 1)))
```

should return #F.

---

2. Ada problem

Write a Ada program to solve the problem described above.

To make the input easier to work with, in this version of the problem, the employees' names will be represented by single letters: "A", "B", etc.

The input will contain 4 lines of 4 letters each, separated by a single space, representing the job assignments. This is followed by a positive integer *N* on a line by itself, indicating the number of employees. This is followed by *N* lines of 4 characters each: the employee's name (a single letter), then three numbers 1 or 0 indicating whether the person is good at answering phones, repairing computers, and fixing the network, in that order; these 4 characters will be separated by a single space.

You are guaranteed that there will be at least one employee and there will be no more than 26 employees.

The output will be one of the two messages "Acceptable" or "Not Acceptable".

Please stick strictly to the input format, as it will make your program easier to test.

Examples:

Input:

```
P D K L
B T R F
S N A C
E G M O
17
A 1 1 1
B 1 1 1
```

```
C 1 1 1
D 1 1 1
E 1 1 1
F 1 1 1
G 1 1 1
J 1 1 1
K 1 1 1
L 1 1 1
M 1 1 1
N 1 1 1
O 1 1 1
P 1 1 1
R 1 1 1
S 1 1 1
T 1 1 1
```

Output:

```
Acceptable
```

Input:

```
Y C V L
D E K S
F G M T
H J P Z
20
A 1 0 0
B 0 1 1
C 1 1 1
D 1 0 0
E 1 0 0
F 1 0 1
G 1 0 1
H 1 0 0
J 1 1 1
K 0 1 1
L 1 0 1
M 0 1 1
P 1 1 1
Q 1 1 0
R 1 0 0
S 0 1 1
T 0 0 1
V 0 1 0
Y 0 1 0
Z 1 1 1
```

Output:

```
Not Acceptable
```

---

3. Prolog problem

Write a Prolog program to solve the problem described above.

Define a predicate *jobs_ok(+Assignment, +Employees)*. *Assignment* is a list of four sub-lists, one for each shift. Each sub-list contains the work assignment for that shift: 2 names for phones, followed by one for computer repair, followed by one for network repair. *Employees* is a list of sub-lists; each sub-list contains an employee's name, followed by three integers (1 or 0) indicating whether the employee is qualified for phones, computers,

and networks, respectively. The predicate should succeed if the job assignment is acceptable and fail if the job assignment is not acceptable.

You may assume that the *Assignment* and *Employees* lists will be in the correct format when the function is called; you do not have to error-check for a non-list or an incorrectly formed list. In your program, you may write and call any additional predicates that are helpful in the computation.

Examples:

```
?- jobs_ok(
   [
        [park, dunbar, khan, lloyd],
        [bashir, trivedi, ramos, franklin],
        [schmidt, nakamura, andrews, chen],
        [evans, goldberg, macmillan, ortiz]],
   [
        [andrews, 1, 1, 1],
        [bashir, 1, 1, 1],
        [chen, 1, 1, 1],
        [dunbar, 1, 1, 1],
        [evans, 1, 1, 1],
        [franklin, 1, 1, 1],
        [goldberg, 1, 1, 1],
        [jefferson, 1, 1, 1],
        [khan, 1, 1, 1],
        [lloyd, 1, 1, 1],
        [macmillan, 1, 1, 1],
        [nakamura, 1, 1, 1],
        [ortiz, 1, 1, 1],
        [park, 1, 1, 1],
        [ramos, 1, 1, 1],
        [schmidt, 1, 1, 1],
        [trivedi, 1, 1, 1]]).
```

should succeed, and

```
?- jobs_ok(
   [
        [young, collins, vieira, liu],
        [davis, ericson, kelly, singh],
        [fitzgerald, gutierrez, martinez, tortorella],
        [hughes, jones, patel, zimmerman]],
   [
        [aziz, 1, 0, 0],
        [blumenthal, 0, 1, 1],
        [collins, 1, 1, 1],
        [davis, 1, 0, 0],
        [ericson, 1, 0, 0],
        [fitzgerald, 1, 0, 1],
        [gutierrez, 1, 0, 1],
        [hughes, 1, 0, 0],
        [jones, 1, 1, 1],
        [kelly, 0, 1, 1],
        [liu, 1, 0, 1],
        [martinez, 0, 1, 1],
        [patel, 1, 1, 1],
        [qureshi, 1, 1, 0],
        [rodriguez, 1, 0, 0],
        [singh, 0, 1, 1],
        [tortorella, 0, 0, 1],
        [williams, 1, 0, 0],
        [vieira, 0, 1, 0],
        [young, 0, 1, 0],
```

```
        [zimmerman, 1, 1, 1]]).
```

should fail.

How to turn in your programs:

- *Give me a paper printout of the text of the program*
- *Give me a paper printout of a sample run of the program (use* script *to capture the input and output)*
- *Copy your program file into the homework directory I have set up for you. (Details on doing this will be in a handout that I will distribute before the first due date.)*

Nancy Tinkham
Computer Science Department, *Rowan University*