Traversal with Prolog

For this programming assignment, we will be writing a program in Prolog that will tell us how to get from one room of a one-story building, to any other room in that building (if it's possible), by telling us all of the rooms we must go through to get to the destination room. In addition to the previous statement, there will be phones ringing in one or more of the rooms. Our prolog program should ONLY tell us how to get to those rooms. If we attempt to go to a room that does not have a ringing phone, the program should not produce any output.

This project consists of one mandatory part, README & Bonus:

1. Prolog code (90%)
2. README.md (10%)
3. Bonus (5%)

Deliverables for Project 3:

1. Fully runnable prolog code (traversal.pl)
2. Project document "markdown" file (README.md)

Notes:

- We will test your results by running your code. If it is not running, mention in README which part is unfinished. If not mentioned in README, points will be given zero.
- All files should be zipped into a single file for submission on Canvas named - (*Project3_LastNameFirstName.zip*).
- Make sure you submit required files only, any kind of missing or extra submission will not be graded after late submission period.

Instructions

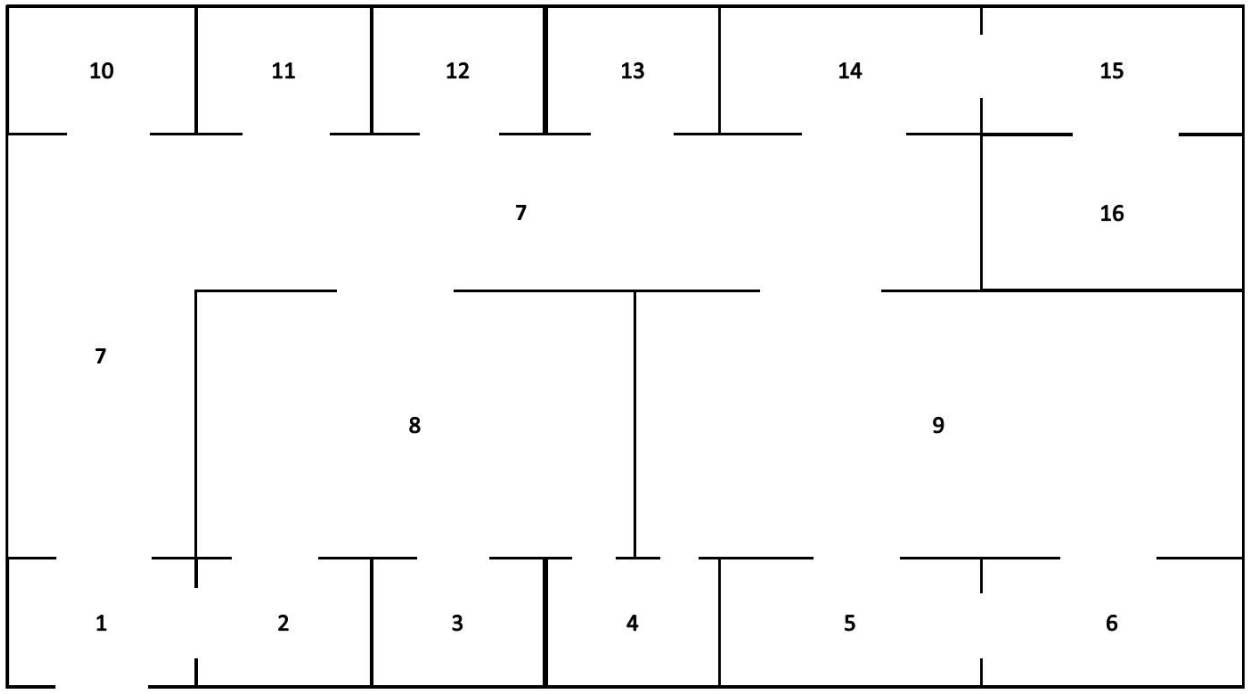
There are a couple of conditions that we should think about to help us frame the problem and hopefully help us understand how we should approach this problem.

1) If I give the program a starting room and an ending room, if there is a phone ringing in the ending room, the program should tell me how to get to that room (which rooms do I need to go through?).
2) Alternatively, if I give the program a starting room and an ending room, and there is no phone ringing in that room, the program should not return any output (actually, prolog should just print "no").
3) If I only give the program a starting room. The program should give me every possible sequence of rooms that I could go through, starting from the starting room that I gave it, to reach any room with a phone ringing.
4) If I only give the program an ending room, if there is a ringing phone in that room, the program should give me every possible sequence of rooms that I could go through (including multiple starting rooms, and all sequences that start with each of those rooms), to reach that room. If there is no phone ringing, it should simply return "no".
5) Finally, if I don't give the program any starting or ending room, it should simply return every possible sequence of rooms I must go through to reach all of the rooms with phones ringing (should be a lot!).

For the bonus, in addition to listing the sequences (steps 1 – 5 above), the program should also display how many rooms we had to traverse. There are two parts to the bonus. They are worth 5 points each. In addition to printing how many rooms we had to traverse you should:

1) Only list the shortest sequence of rooms to get to a given room with a phone ringing. That means if there are 3 phones ringing, you should only display 3 sequences; the shortest sequence to get to each of the rooms with a phone ringing.
    a. TIP: This only has to work for conditions where you AT LEAST know the starting room (so #4 above doesn't have to do this correctly).
2) Only list the longest sequence of rooms to get to a given room with a phone ringing.
    a. TIP: same as tip for the previous bonus

Below is a picture of the one-story building you MUST base your program off of. In the graphic below, every opening represents a door between one room and another.



In this building, I can go from room 1 to rooms 2 and 7. I can also go from room 7 to many rooms, but I CANNOT go from room 7 to room 16. If I need to get to room 16, I must first go from 7 to 14, then from 14 to 15, then from 15 to 16. So one sequence of rooms prolog should return if I wanted to know how to get from room 1 to room 16 (and if there happened to be a phone ringing in room 16) is: 1, 7, 14, 15, 16.

Below are samples of input you might give to the program and what it should return. **(No Bonus)**

```
GNU Prolog console
File  Edit  Terminal  Prolog  Help
GNU Prolog 1.4.4 (64 bits)
Compiled Apr 23 2013, 16:41:01 with x86_64-w64-mingw32-gcc
By Daniel Diaz
Copyright (C) 1999-2013 Daniel Diaz
| ?- consult('D:/Dropbox/auburn_faculty/prog_languages/spring 17/assgn/prog_assgn3/maze_with_phone_key_nbon
compiling D:/Dropbox/auburn_faculty/prog_languages/spring 17/assgn/prog_assgn3/maze_with_phone_key_nbonus.p
D:/Dropbox/auburn_faculty/prog_languages/spring 17/assgn/prog_assgn3/maze_with_phone_key_nbonus.pl compiled

yes
| ?- path_to_phone(1,5,My_Path).

My_Path = [1,2,8,4,9,5] ? ;

My_Path = [1,2,8,4,9,6,5] ? ;

My_Path = [1,2,8,7,9,5] ? ;

My_Path = [1,2,8,7,9,6,5] ? ;

My_Path = [1,7,8,4,9,5] ? ;

My_Path = [1,7,8,4,9,6,5] ? ;

My_Path = [1,7,9,5] ? ;

My_Path = [1,7,9,6,5] ? ;

no
| ?- |
```

In the example above, we wanted to know every sequence of rooms we had to go through to get from room 1 to room 5. There also happened to be a phone ringing in room 5.

```
no
| ?- path_to_phone(1,4,My_Path).

no
| ?-
```

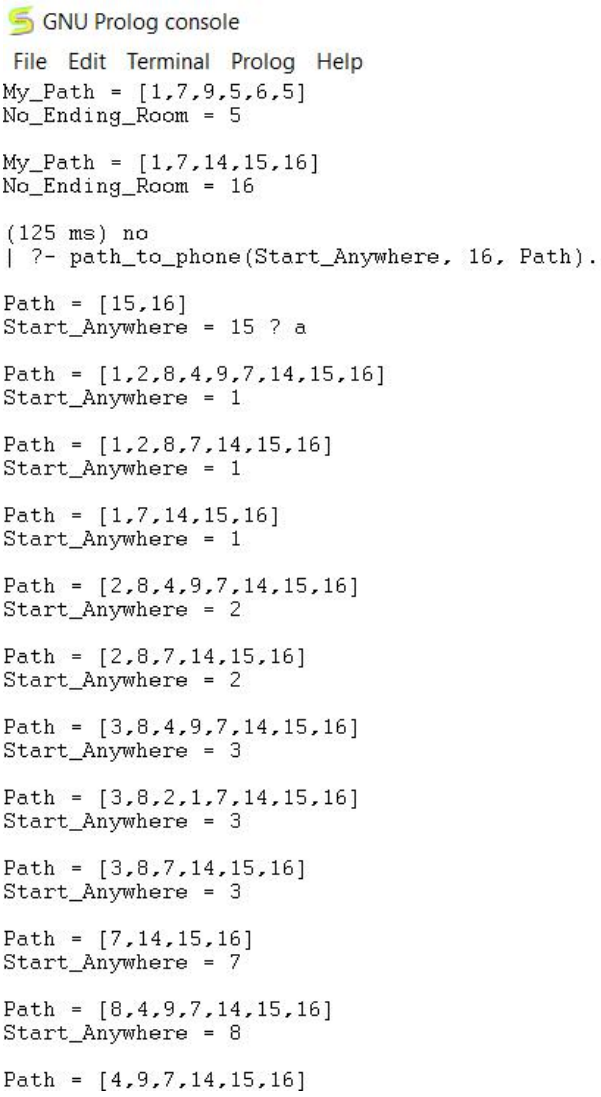
In the example above, we wanted to know every sequence of rooms we had to go through to get from room 1 to room 4. Although there is a way we could reach room 4 from room 1, there was no phone ringing in room 4, so Prolog just told us "no".

GNU Prolog console

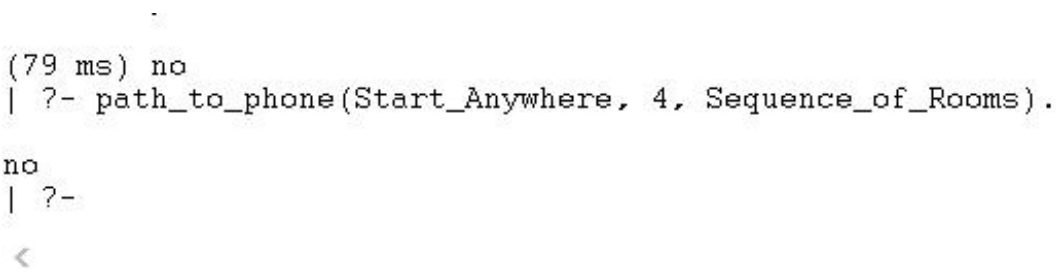
File Edit Terminal Prolog Help

```
no
| ?- path_to_phone(1,No_Ending_Room, My_P

My_Path = [1,2,8,4,9]
No_Ending_Room = 9 ? a

My_Path = [1,2,8,4,9,5]
No_Ending_Room = 5

My_Path = [1,2,8,4,9,6,9]
No_Ending_Room = 9

My_Path = [1,2,8,4,9,6,5]
No_Ending_Room = 5

My_Path = [1,2,8,4,9,6,5,9]
No_Ending_Room = 9

My_Path = [1,2,8,4,9,7,9]
No_Ending_Room = 9

My_Path = [1,2,8,4,9,7,14,15,16]
No_Ending_Room = 16

My_Path = [1,2,8,4,9,5,9]
No_Ending_Room = 9

My_Path = [1,2,8,4,9,5,6,9]
No_Ending_Room = 9

My_Path = [1,2,8,4,9,5,6,5]
No_Ending_Room = 5

My_Path = [1,2,8,7,9]
No_Ending_Room = 9

My_Path = [1,2,8,7,9,5]
No_Ending_Room = 5

My_Path = [1,2,8,7,9,6,9]
No_Ending_Room = 9

My_Path = [1,2,8,7,9,6,5]
No_Ending_Room = 5
```

In the example above, we gave a starting room (room 1), but no ending room. There were phones ringing in room 5, 9, and 16. There were a lot of sequences. These are just some of them.



In the example above, we did not give the program a starting room, and told it to try to get to room 16. There were a lot of sequences. These are just some.



In the example above, we did not give a starting room, but we did give an ending room. Since there is no phone ringing in room 4, Prolog just told us no.

Below are samples of input you might give to the program and what it should return for Bonus. **(Bonus) For all bonus objectives, they only have to work correctly as long as you provide a starting room.**

```
| ?- min_path_to_phone(1,Any_Room, Sequence_of_Rooms, How_Many_Rooms).

Any_Room = 5
How_Many_Rooms = 3
Sequence_of_Rooms = [1,7,9,5] ? a

Any_Room = 9
How_Many_Rooms = 2
Sequence_of_Rooms = [1,7,9]

Any_Room = 16
How_Many_Rooms = 4
Sequence_of_Rooms = [1,7,14,15,16]

yes
| ?- |
```

In the example above, we tell Prolog to start at room 1, and give is the shortest sequence of rooms to get to any room with a phone ringing. Recall that *there are phones in room 5, 9, and 16*. Note that it ONLY returned 3 answers, 1 for each of the rooms.

```
yes
| ?- max_path_to_phone(1,16,Sequence_of_Rooms, Distance).

Distance = 8
Sequence_of_Rooms = [1,2,8,4,9,7,14,15,16]

yes
| ?-
```

In the example above, we tell prolog to search for the largest sequence of rooms we have to go through to get from room 1 to room 16 (which has a phone ringing in it).

FINAL NOTE:

Your program MUST use the room layout provided above. Aside from that requirement, there is **NO GUARANTEE** that phones will always be ringing in rooms 5, 9, and 16. This is arbitrary, and your program should work **REGARDLESS** of which rooms have phones ringing in them. You may choose to use those values to test your program if you want to ensure that the output is the same as the example screenshots above.

The screenshots were taken using GNU prolog for windows. You can also download the program yourself for Mac or Windows. For this assignment your program **MUST WORK in GNU Prolog or SWI-Prolog**. There are other variants of Prolog that behave slightly differently and use different syntax. We will grade using GNU Prolog or SWI-Prolog. Mention in README, How to run your program successfully.

Late submission penalty:

- Ten percent (10%) penalty per day for late submission. For example, an assignment submitted after the deadline but up to 1 day (24 hours) late can achieve a maximum of 90% of points allocated for the assignment.
- Assignment submitted more than one day (24 hours) after the deadline will not be graded.

Rebuttal period:

- You will be given a period of 4 days to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.

(Important) There should be no collaboration among students. Each student should NOT share any project code with any other students. Collaborations among students in any form will be treated as a serious violation of the University's academic integrity code.