CS 359 – Programming Paradigms

PEX 3 – PERT

Preliminary Submission Due: Taps, Lesson M28, Wednesday, October 31

Final Submission Due: Taps, Lesson M32, Tuesday, November 13

Help Policy:

AUTHORIZED RESOURCES: Any, except another cadet's program.
NOTE:

- Never copy another person's work and submit it as your own.
- Do not jointly create a program unless explicitly allowed.
- You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook).
- **DFCS will recommend a course grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.**

Documentation Policy:

- You must document all help received from any source other than your instructor.
- The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
- If no help was received on this assignment, the documentation statement must state "NONE."
- If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why.
- **Vague documentation statements must be corrected before the assignment will be graded and will result in a 5% deduction on the assignment.**

Turn-in Policies:

- On-time turn-in is at the specific time listed above.
- Late penalties accrue at a rate of 25% per 24-hour period past the on-time turn-in date and time. The late penalty is a cap on the maximum grade that may be awarded for late work.
- There is no early turn-in bonus or extra credit for this assignment.

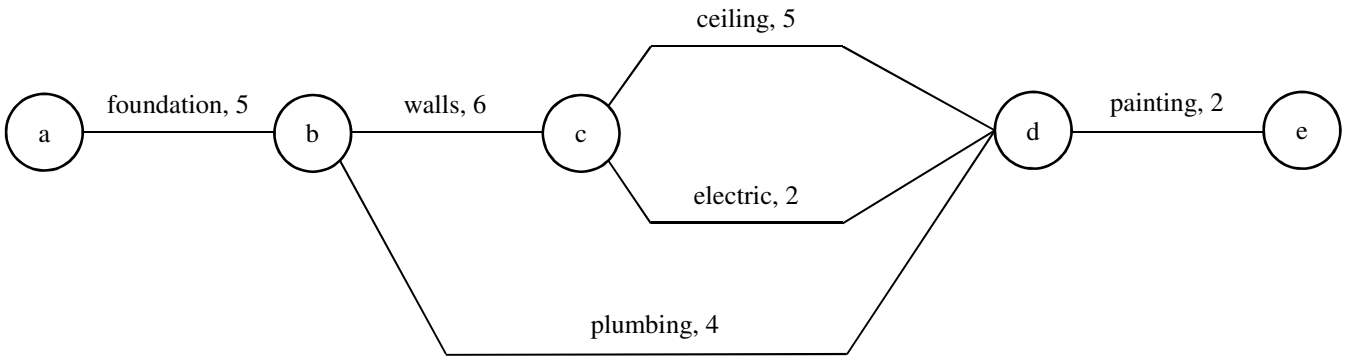
1. OBJECTIVES

- Expand your range of programming language competence
- Be able to define and use Prolog facts and rules
- Be able to use the Prolog list structure
- Be able to create, test, and debug a medium sized Prolog program
- Be able to represent a PERT chart using Prolog and obtain relevant information from it

2. BACKGROUND

Your task for this exercise is to write a collection of Prolog fact and rule statements to implement a PERT chart. Program Evaluation and Review Technique is a project management technique designed to analyze and represent the tasks involved in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project. A PERT chart is a tool that facilitates this analysis. Consecutive events in a PERT chart are linked by activities. The events are presented in a logical sequence and no activity can commence until its immediately preceding event is competed.

For example, consider the following diagram with tasks involved in building a house:



The idea is that it takes five days to pour the foundation, and only after the foundation is done can work begin on the walls and plumbing. It takes six days to build the walls, and only after the walls are finished can work begin on the ceiling and electric. Etc.

A PERT ***event*** is a point that marks the start or completion of one or more activities. It consumes no time and uses no resources. In the diagram above, a, b, c, d, and e are events. One event must be designated as the ***start event*** and one event must be designated as the ***end event*** (a and e in the diagram above).

A PERT ***activity*** is a task necessary to move from one event to another. An activity does consume time and resources. In the diagram above, foundation, walls, ceiling, electric, plumbing, and painting are activities. The numbers represent the time required to complete the activity (days in this example; resources are ignored in this exercises).

A ***predecessor event*** immediately precedes an activity and a ***successor event*** immediately follows an activity. In the diagram above, the plumbing activity's predecessor event is b and successor event is d.

A ***path*** is a series of activities that connect the start event to the end event. The ***critical path*** is the longest possible continuous path from the start event to the end event. The critical path determines the total time required for the project. In the diagram above, [foundation, walls, ceiling, painting] is the critical path.

The ***float time*** of an activity is a measure of the excess time available to complete an activity. That is, it is the amount of time an activity can be delayed without causing delay in any subsequent activities. A ***critical activity*** is an activity with a float time of zero. In the diagram above, plumbing's float time is 7.

The ***lead time*** of an activity is the time prior to project completion by which an activity must be completed. In the diagram above, the lead time for completing the walls is 7.

3. PRELIMINARY EXERCISE

For the preliminary exercise you will create two PERT charts, drawing diagrams and writing their representations in Prolog, and create a **`path/3`** rule. The first PERT chart should be a smaller chart similar in size to the sample in this document. It should have at least three possible paths between the start event and end event with one being the critical path. It should describe a project you are familiar with, though it will likely describe a trivial project such as washing your hair or preparing a meal.

The second PERT chart should be a larger chart, perhaps three to four times the size of the sample in this document. It should describe a lengthier project, have multiple paths between the start and end events, and have more than one critical path, each with several activities distinct from the other critical path.

The **`path/3`** rule will be used to determine paths between events in your PERT charts. Specifically, given the sample PERT chart in this document, the rule should behave as follows:

```
?- path( a, e, [foundation, walls, ceiling, painting] ).
true ;
false.

?- path( a, e, Path ).
Path = [foundation, walls, ceiling, painting] ;
Path = [foundation, walls, electric, painting] ;
Path = [foundation, plumbing, painting] ;
false.

?- path( c, c, Path ).
Path = [] ;
false.
```

4. PRELIMINARY EXERCISE – HELPFUL HINTS

There are many ways a PERT chart could be represented in Prolog. For the purposes of this exercise, it will be very helpful if everyone uses the same representation. Thus, the format of the following facts that represent the sample PERT chart in this document should be used when representing your PERT charts:

```
event( a ).                     activity( a, b, foundation, 5 ).
event( b ).                     activity( b, c, walls, 6 ).
event( c ).                     activity( b, d, plumbing, 4 ).
event( d ).                     activity( c, d, ceiling, 5 ).
event( e ).                     activity( c, d, electric, 2 ).
                                activity( d, e, painting, 2 ).
start( a ).
end( e ).
```

Do not forget to include the appropriate fact header comments for each of the above facts.

5. PROGRAMMING EXERCISE

Implement the remaining functionality described in the background section of this document. Specifically, you are to write Prolog rules for **`time/2`**, **`longer_path/3`**, **`critical_path/2`**, **`float_time/2`**, **`critical_activities/0`**, and **`lead_time/2`**.

The following demonstrates the use of each of these rules:

```
?- time( [foundation, walls, ceiling, painting], 18 ).
true.
?- time( [foundation, walls, ceiling, painting], Time ).
Time = 18.
?- time( [plumbing, painting], Time ).
Time = 6.
?- time( [], Time ).
Time = 0.
?- longer_path( a, e, 16 ).
true ;
false.
?- longer_path( a, e, 20 ).
false.
?- longer_path( b, d, 4 ).
true ;
true ;
false.
?- critical_path( [foundation, walls, ceiling, painting], 18 ).
true ;
false.
?- critical_path( Path, Time ).
Path = [foundation, walls, ceiling, painting],
Time = 18 ;
false.
?- float_time( plumbing, 7 ).
true ;
false.
?- float_time( plumbing, Time ).
Time = 7 ;
false.
?- float_time( walls, Time ).
Time = 0 ;
false.
?- critical_activities.
foundation
walls
ceiling
painting
false.
?- lead_time( walls, 7 ).
true ;
false.
?- lead_time( walls, Time ).
Time = 7 ;
false.
```

6. PROGRAMMING EXERCISE – HELPFUL HINTS

The required **`critical_path`** rule has two parameters, the path and the time. You will find it useful to define another **`critical_path`** rule with four parameters, the start event, end event, path, and time.

Make use of the built-in predicate **`not`**. Specifically, you might use it in combination with the **`longer_path`** rule: **`not( longer_path( Start, End, Time ) )`**

7. SUBMISSION REQUIREMENTS

- **INCLUDE AN APPROPRIATE DOCUMENTATION STATEMENT (PRELIM/PEX) AT THE TOP OF THE SOURCE FILE CONTAINING YOUR RULE DEFINITIONS!**
- **All function names must exactly match those in this document as this will be auto-graded.**
- For both the preliminary exercise and programming exercise, you must create and submit three separate Prolog source files:
  - **`Lastname0.pl`** – This file will contain your rule definitions.
  - **`Lastname1.pl`** – This file will contain your fact definitions for the smaller PERT chart.
  - **`Lastname2.pl`** – This file will contain your fact definitions for the larger PERT chart.
  - Note: Each of the above file names should use your own last name in place of "**`Lastname`**".
  - Note: There is no "**`Prelim3`**" or "**`PEX3`**" or "**`PERT`**" or any other prefix or suffix in those file names … just your own last name and the appropriate single digit.
- For the preliminary exercise, you must also submit your diagrams in electronic format.
  - The preferred format is **`pdf`**. If you need assistance scanning a hand-drawn diagram into **`pdf`**, please contact your instructor.
  - If you plan to submit anything other than **`pdf`**, please confirm the format with your instructor – if your file cannot be opened, it will not be correct!
- For the programming exercise, you must re-submit the Prolog source code for your diagrams.
- Create a folder on your system with everything you would like to be graded and *name this folder* with *your own last name*.
- Zip the entire folder to a file with the name **`Lastname.zip`**, using your own last name.
  - Accomplish this by right-clicking on the folder and choosing Send To → Compressed (zipped) folder.
  - Note there is no "PEX3" or "Prelim3" or "PERT" or any other prefix or suffix in that file name … just your own last name.
- Use the website to submit a **single zip file** containing everything you would like to be graded.

CS 359 – PEX 3 Prelim – Grade Sheet Name:_____

Criteria		Points Earned	Points Available
Smaller PERT chart – diagram			**4**
Smaller PERT chart – Prolog			**4**
Larger PERT chart – diagram			**6**
Larger PERT chart – Prolog			**6**
path/3 rule			**10**
	Subtotal:		**30**
Adjustments	**All code meets specified standards:**		**– 3**
	Vague/Missing Documentation:		**– 2**
	Submission Requirements Not Followed:		**– 2**
	Late Penalties:		**25/50/75%**
	Total w/adjustments:		

Comments from Instructor:

CS 359 – PEX 3 – Grade Sheet Name:_____

Criteria		Points Earned	Points Available
time/2 rule			**8**
longer_path/3 rule			**8**
critical_path/2 rule			**10**
float_time/2 rule			**10**
critical_activities/0 rule			**8**
lead_time/2 rule			**8**
Overall quality of software design			**8**
	Subtotal:		**60**
Adjustments	**All code meets specified standards:**		**– 6**
	Vague/Missing Documentation:		**– 3**
	Submission Requirements Not Followed:		**– 3**
	Late Penalties:		**25/50/75%**
	Total w/adjustments:		

Comments from Instructor: