CMPE 224
Fall 2021
Programming Homework 3

This assignment is due by 23:59 on Friday, 3 December 2021.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE224-HW3-OfficeHour1: 19 Nov, 07:00-09:00 PM, Zoom ID: 984 7950 5886
   - CMPE224-HW3-OfficeHour2: 26 Nov, 07:00-09:00 PM, Zoom ID: 972 7018 1723

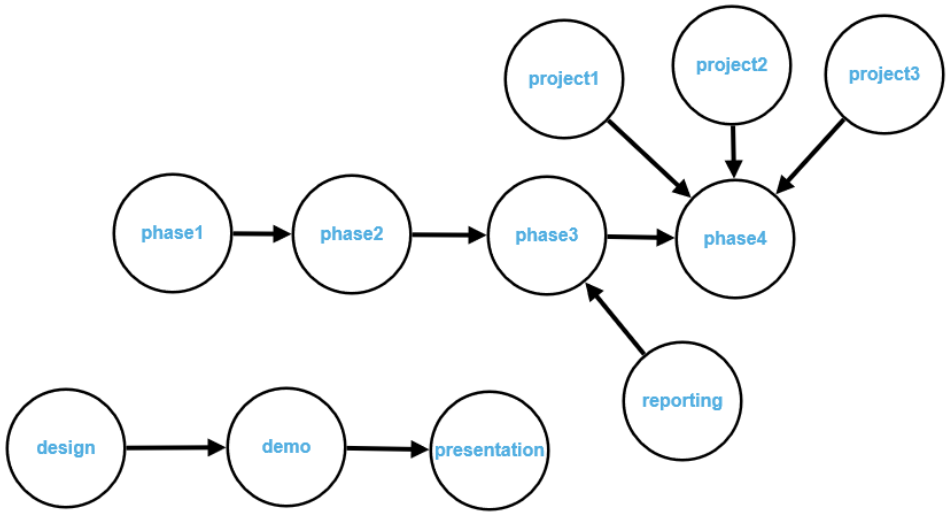
Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

Suppose you are the project manager of a software company and the workload of the company consists of *N* tasks for a month. To complete the workload, you should finish all tasks at the end of the month and there are some tasks with prerequisites. A prerequisite is a task that must completed before starting in a second task. Some tasks may require more than one prerequisite and some tasks may require any prerequisite. And you can complete 3 jobs in a week because there are 3 teams.

Some of these tasks are independent – it doesn't really matter in what order you do the 3 projects, for example. But other tasks do depend on one another, **and you need to complete the dependent tasks on different weeks because you cannot start one task before you complete its prerequisite.** You have to design before you create a demo, and you have to create a demo before you prepare a presentation. You need to do reporting before your third phase, and you have to finish 3 projects before the fourth phase. The question is, what order should you do your tasks in?

Visually, the dependencies between tasks are shown below:



This graph is given in `sampleinput.txt` file, given below. The file contains 11 tasks, and 9 dependencies:

```
11
project1
project2
project3
phase1
phase2
phase3
phase4
reporting
demo
presentation
design
3 4
4 5
5 6
8 9
10 8
7 5
0 6
```

```
1 6
2 6
-1 -1
```

In the input file, the first line contains the number N of vertices. Each of the following N lines is a string, which you should store as the names of the N vertices. Each vertex is given an ID, starting from 0.

Following the vertices, there will be a series of lines containing two numbers: `v1 v2`. These are edges: the first number is the index of the starting vertex, and the second number is the index of the target vertex. The list of edges ends either at the end of the file, or when it reads a line "-1 -1".

Note: You will be given different input files to work with. You are recommended to work with smaller graphs first, and then try on larger graphs. **And also, there may be more than one solution for the given sample input because there may be more than one topological order. On VPL, there will be more than one test cases according to different topological orders.**

```
%java Company sampleinput.txt
Enter choice (0: Exit, 1: List schedule, 2: Check order): 1
Week 1: project1 phase1 reporting
Week 2: project2 phase2 design
Week 3: project3 phase3 demo
Week 4: phase4 presentation
Enter choice (0: Exit, 1: List schedule, 2: Check order): 2
Enter first task: project1
Enter second task: project2
You should do project1 before project2.
Enter choice (0: Exit, 1: List schedule, 2: Check order): 2
Enter first task: phase2
Enter second task: project1
You should do phase2 after project1.
Enter choice (0: Exit, 1: List schedule, 2: Check order): 2
Enter first task: phase4
Enter second task: presentation
You should do phase4 and presentation on the same week.
Enter choice (0: Exit, 1: List schedule, 2: Check order): 0
```

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You can test your Java source files on available Moodle VPL environment to ensure your solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input.
- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%30): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%40): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%15): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%10): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Friday, 3 December 2021.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-----------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-----------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//---------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//---------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TAs, Bedrettin Çetinkaya, Deniz Merve Gündüz and İbrahim İleri. Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructors Tolga Çapın and Ulaş Güleç for help.