Operating systems 2
Projects' General Guidelines

- Any Teams who plagiarize (copy/paste) will be subject to penalties including drop course.
- All projects must have organized documentation:
  - project description.
  - what you have actually did.
  - team members role.
  - code documentation.
- The team could have 5 - 7 members (no exceptions).
- All projects should be implemented in JAVA and should include GUI

جامعة حلوان HELWAN UNIVERSITY	Operating Systems 2 Projects	كلية الحاسبات والذكاء الاصطناعى FACULTY OF COMPUTING AND AI

Project #1: Sleeping TA

A university computer science department has a teaching assistant (TA) who helps undergraduate students with their programming assignments during regular office hours. The TA 's office is rather small and has room for only one desk with a chair and computer. There are three chairs in the hallway outside the office where students can sit and wait if the TA is currently helping another student. When there are no students who need help during office hours, the TA sits at the desk and takes a nap. If a student arrives during office hours and finds the TA sleeping, the student must awaken the TA to ask for help. If a student arrives and finds the TA currently helping another student, the student sits on one of the chairs in the hallway and waits. If no chairs are available, the student will come back at a later time.

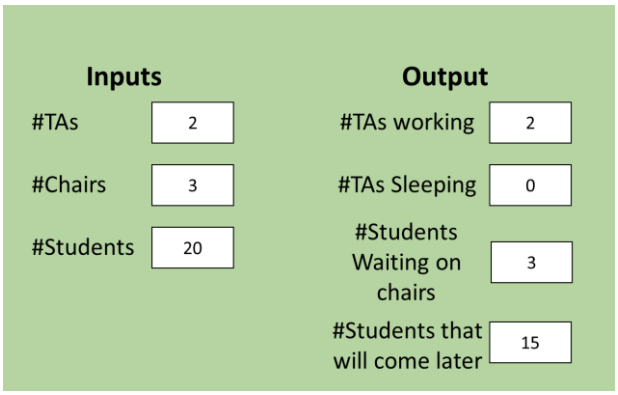
Using JAVA threads, mutex locks, and semaphores, implement a solution that coordinates the activities of the TA and the students. The number of disks in the TA's room (aka no. of TAs), number of chair for waiting students and number of students that have questions should be provided as an input to your program.

Problem Modeling:

- Students are modeled as Threads
- TA(s) and Chairs are modeled as Mutex/semaphore
- Correct logic
    - Student tries to enter TA(s) room
    - If no TA available student try to wait in one of the chairs
    - If no chair is available, student will leave and come later. (s)he will repeat the same logic then.

GUI

- Input:
    - #students,
    - #chairs,
    - #TA(s)
- Output:
    - #TAs working,
    - #TAs Sleeping,
    - #Students Waiting on chairs,
    - #Students that will come later
- Note that updates should be in real time.



جامعة حلوان HELWAN UNIVERSITY	Operating Systems 2 Projects	كلية الحاسبات والذكاء الاصطناعى FACULTY OF COMPUTING AND AI

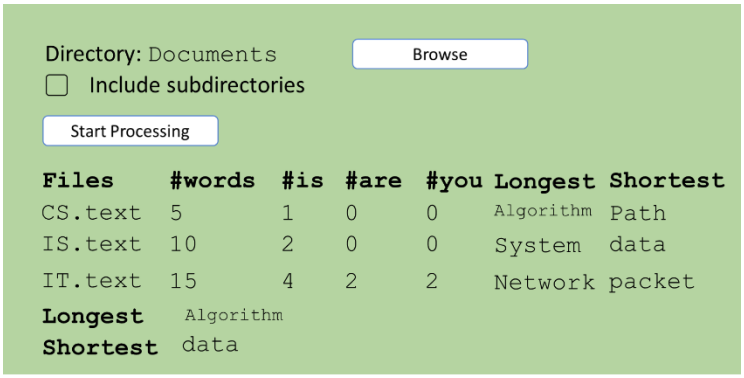
Project #2: Word statistics

Write a program that reads all text files form a specific directory and return word statistics (number of words per file/directory, longest word, shortest word, number of “is”, “are” and “you”).

- The program should have a simple GUI
- The input of the program is a directory
  - It should then search for all text files that reside in that directory
  - There should be an option to check for text files in subdirectories
- While the program is processing text files, it should display them (file names) and the up-to-date statistics. (Statistics should be updated in run-time)

Problem Modeling

- Mian thread identify text files in directory and its subdirectory (one or two level) and show them in GUI
- Each thread explores one or more text files
  - No. of threads is based on the number of processors (core)
- Each thread should send updates to GUI



GUI

- Input
  - Directory path (or selection via browse button)
  - Checkbox for including subdirectories
- Output
  - In table form:
    - #words
    - #is
    - #are
    - #you
    - Longest word per file
    - Shortest word per file
    - Longest word per directory
    - Shortest word per directory

- Note that updates should be in real time.

جامعة حلوان HELWAN UNIVERSITY	Operating Systems 2 Projects	كلية الحاسبات والذكاء الاصطناعي FACULTY OF COMPUTING AND AI

Project #3: Rat in a Maze Problem

A Maze is given as $N \times N$ binary matrix of blocks where source block is the upper left most block i.e., maze [0][0] and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from source and must reach the destination. The rat can move only in two directions: forward and down. In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination.

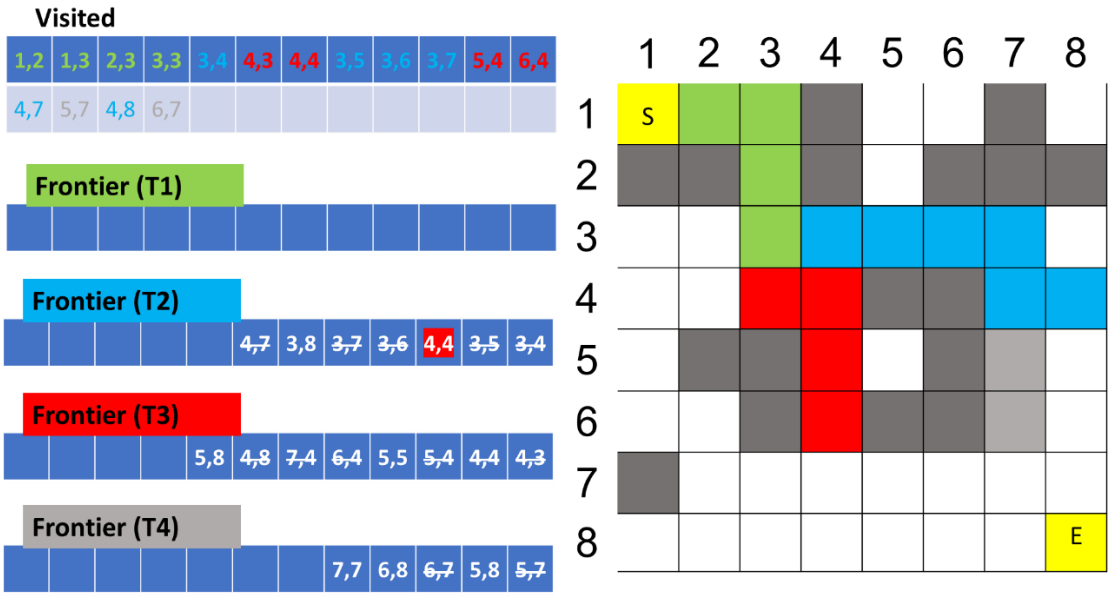
Use multi-threading to solve this problem.

You should design a multithreaded JAVA program with the following features:

- You should enter the dimensions of the maze, then a grid is generated.
- You should use the grid to specify dead blocks on runtime

Problem Modeling

- The maze is modeled as a matrix of dimension $N * N$
  - Start is the upper left cell
  - End is the lower right cell
  - The mouse only moves forward or down
- If thread finds two possible directions it will continue in one and create new thread for the second possible direction.
  - The number of threads should be limited based on the number of processors



GUI

- Input:
  - Maze size (N)
- Output
  - Maze(s) shows rat path in Realtime
- Note: Update should be in real time.

Project #4: N Queen Problem

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other by being in the same row, column or diagonal. Use multi-threading to solve this problem.

Problem modeling

- N queen board are modeled as an $N \times N$ matrix
- Main thread creates N threads or N/x (where x is number of processors) that explore solution space for the board.
    - Each thread will start given the position of the queen in the first row
    - For large N, a thread may handle to branches of the Graph the modeled the search space

GUI:

- Input
    - N board dimension
- Output
    - Board(s) showing search space exploration for solution
- Note: GUI should be updated in real time.

Project #5: Make a square

Make a square with size 4X4 by using 4 or 5 pieces. The pieces can be rotated or flipped, and all pieces should be used to form a square. Example sets of pieces.

There may be more than one possible solution for a set of pieces, and not every arrangement will work even with a set for which a solution can be found. Examples using the above set of pieces…
Rotate piece D 90 degree then flip horizontal {R 90 + F H}

Input:

The first line contains number of pieces. Each piece is then specified by listing a single line with two integers, the number of rows and columns in the piece, followed by one or more lines which specify the shape of the piece. The shape specification consists of 0 or 1 character, with the 1 character indicating the solid shape of the puzzle (the 0 characters are merely placeholders). For example, piece A above would be specified as follows:

2 3
111
101

Output:

Your program should report all solutions, in the format shown by the examples below. A 4-row by 4-column square should be created, with each piece occupying its location in the solution. The solid portions of piece #1 should be replaced with `1' characters, of piece #2 with `2' characters.

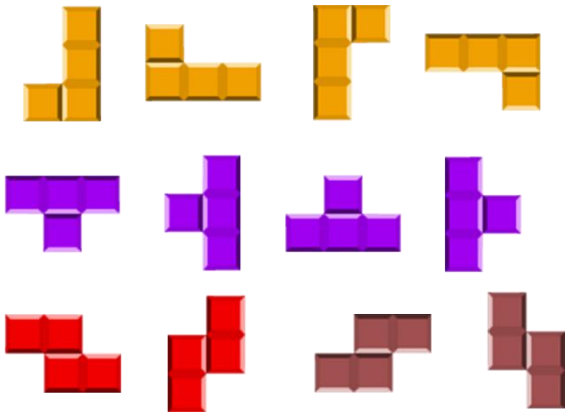
Sample output that represents the figure above could be:

1112
1412
3422
3442

For cases which have no possible solution simply report "No solution possible".
You must provide many sample inputs (many text files) during the discussion time, to test your project on many samples. Each text file represents one problem to be solved.

Problem modeling:

- Square has a fixed dimension of 4×4
- User choose pieces form the standard Tetris pieces
- Each piece has different possible rotation (no flipping is allowed)



- One possible solution is to let a main thread choose one of the pieces
    - Each thread will search for the solution space given a specific rotation for the selected piece
    - The main thread waits till the thread finishes and try another piece if no solution is found

GUI:

- Input:
    - Users choose number of pieces of each type
- Output
    - Display what is happening in each thread on different boards in Realtime
- Note: GUI should be updated in real time.