General Rules for Algorithm  
 final project delivery

**Please make sure you read this WHOLE FILE carefully. No excuse for not following any of the mentioned points.**

**GENERAL RULES:**

* **Avoid cheating or copying tasks. Do your project YOURSELF. Any detected cheating or copying case will be PUNISHED according to the Honor Code Policy.**
* **You MUST bring the documentation PRINTED with the entire code and DETAILED analysis.**
* **You should test the test cases before your delivery. They are similar to delivery ones.**
* **ALL team members are assumed to know the project very well and ready to do any modification in it**
* **No code updates are allowed during delivery.**
* **You MUST commit with the testcases input and output formats given to you.**
* **Place your FINAL COMPLETE code in a ZIPPED folder before your delivery on a FLASH memory in a folder named T#.**

# Image Quantization

* You must output for each SAMPLE case in your interface:
  + # of DISTINCT colors
  + MST Total
* You must output for each COMPLETE case in your interface:
  + MST Total
  + Execution time in **MILLISECONDS & SECONDS**
* Time limits for complete cases:
  + Small cases: 1 MIN
  + Medium cases: 3 MINs
  + Large cases: 6 MINs
* You MUST commit with the format of required Input and Output of the test cases.
* Documentation with the code and **DETAILED COMPLEXITY ANALYSIS**

# N Puzzle

* You MUST commit with the format of required Input and Output of the testcases.
* The project should support the option to run using Manhattan or Hamming.
* Manhattan complexity shouldn't exceed O(N^2)
* A\*: Extracting min from the opened list should be less than O(N) (e.g. logarithmic or constant time)
* A\*: Search in closed or opened lists should be less than O(N) (e.g. logarithmic or constant time)
* A\*: if a node is found in the closed set and its cost is better, then, it should be added to the open set
* You must output for each SAMPLE or COMPLETE case in your interface:
  + Solvable or not
  + If solvable, display the board at each movement till the final solution to simulate the shortest sequence of moves till you solve the puzzle.
  + The fewest number of moves required at the end to re-arrange the puzzle
* You must output for each COMPLETE case in your interface:
  + Execution time in **MILLISECONDS & SECONDS**
* Time limits for complete cases:
  + Solvable using Manhattan only: MAX 10 SECONDS
  + Solvable using Manhattan & Hamming: MAX 15 SECONDS
  + Very large: 2 MINs
* Documentation with the code and **DETAILED COMPLEXITY ANALYSIS**

# Intelligent Scissors

* You must output for each SAMPLE case in your interface:
  + The constructed graph (check details in the testcases).
  + Shortest path value for the shortest path.
* You must output for each SAMPLE & COMPLETE case in your interface:
  + Draw final shortest path between an anchor point and next new anchor point (mouse click).
  + Draw shortest path between an anchor point and a free point (mouse move).
  + Close lasso by drawing the shortest path between the final anchor point (occurred on double mouse click) and the original starting anchor point.
* You must output for each COMPLETE case in your interface:
  + RUN (Draw connected paths) -> FAST AND SMOOTH
  + Close Lasso
  + Execution time in **MILLISECONDS & SECONDS**
* Time limits for complete cases:
  + Graph construction (Time limits specified in the testcases document).
  + No time to draw the path (FAST & SMOOTH) on each mouse click or mouse move.
* You MUST commit with the format of required Input and Output of the test cases.
* Documentation with the code and **DETAILED COMPLEXITY ANALYSIS**

# Small World Phenomenon

* The project should support the option to run with and without the optimization idea.
* You need to print one chain of films between two actors A, B (i.e. in the solution document you have it's written 1 or 7 => 4, you should print either 1 => 4 or 7 => 4)
* You must output for each SAMPLE & COMPLETE case in your interface:
  + The degree of separation between the two actors/actresses.
  + The relation strength (total number of common movies) between them.
  + The shortest chain of movies that leads the first actor/actress to the second.
* You must output for each COMPLETE case in your interface:
  + Execution time in **MILLISECONDS & SECONDS**
* Time limits for complete cases:
  + Small: No time
  + Medium, large & extreme: < 3 MINs
* You MUST commit with the format of required Input and Output of the test cases.
* Documentation with the code and **DETAILED COMPLEXITY ANALYSIS**

# WordNet Semantics

* You must output for each SAMPLE & COMPLETE case in your interface:
  + Semantic Relations **File**: Distance and query results of each shortest common ancestor of each pair of nouns
  + Outcast Detection **File**: The outcast (odd) noun from each case of nouns
* You MUST commit with the format of required Input and Output of the test cases.
* OUTPUT MUST be in the same format of the given output file of each testcase (OTHERWISE will not be accepted).
* You must output for each COMPLETE case in your interface:
  + Execution time in **MILLISECONDS & SECONDS**
* Time limits for complete cases:
  + Given with each testcase
* Documentation with the code and **DETAILED COMPLEXITY ANALYSIS**