**Project 4 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item #.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | oneSweep.m | Total = 10 | Total = 4 |
| 1 | Use a nested loop to go through all cells in the matrix. |  | 1 |
| 2 | outM is a 0-1 matrix of the same size as inM. | 1 |  |
| 3 | For every cell, use the initial state when computing the transition. | 1 |  |
| 4 | Slice inM to get a cell and its neighbors (1S). Correct slicing (2C, -1C for each error). | 2 | 1 |
| 5 | Compute the number of alive cells in the sliced matrix. | 1 |  |
| 6 | Update the state of the cell. (2C, -1C for each error) | 2 |  |
|  | drawState.m |  |  |
| 7 | Use a nested loop to go through all cells in the matrix. |  | 1 |
| 8 | Correctly draw a dot or asterisk for each cell. (2C, -1C for each error) | 2 |  |
| 9 | Display the step number in the title. |  | 1 |
|  | gameOfLife.m |  |  |
| 10 | Correctly call oneSweep and drawState.  **NOTE**: Since the skeleton of this script is in the writeup, we only give one point for calling the two functions. | 1 |  |
| PROBLEM 2 | circuit.m | Total = 1 | Total = 0 |
| 11 | Create the 3\*3 matrix. Apply matrix left division to solve for I\_1, I\_2, I\_3 and display them.  **NOTE**: We use only 1 correctness point for this question. A student does not need to get the correct answer to get this point. Just create the matrix and apply left division is worth the point. | 1 |  |
| PROBLEM 3 | pointillize.m | Total = 11 | Total = 5 |
| 12 | Halting execution with an error message when the number of blocks in either or both dimensions is <10 (1S), and when ani is 1 and the number of blocks in either or both dimensions is >80 (1S). |  | 2 |
| 13 | Correctly compute nrb and ncb. | 1 |  |
| 14 | Correctly crop the image so that its number of rows and columns are each a multiple of n. (2C: -1C for each error)  **NOTE**: Crop from the four edges of the image (data) evenly so that the middle data are kept; If the number of rows (columns) of pixels to crop is odd, then crop more from the bottom (right) than from the top (left), as is specified in the writeup. | 2 |  |
| 15 | colr and gray are type double matrices of correct size. | 1 |  |
| 16 | Use a nested loop to go through all blocks |  | 1 |
| 17 | Correctly slice the matrix to get each block. | 1 |  |
| 18 | Correctly compute the value of colr. | 1 |  |
| 19 | Correctly compute the value of gray. | 1 |  |
|  | drawDots.m |  |  |
| 20 | Use a nested loop to go through all dots. |  | 1 |
| 21 | Use correct color for each dot.  **NOTE**: A block whose gray intensity is >=t is shown as a disk in its gray intensity, else shown in color. | 1 |  |
| 22 | Use corret radius for each dot. | 1 |  |
| 23 | Call DrawDiskNoBorder to draw the dots in correct position. | 1 |  |
|  | animateDots.m |  |  |
| 24 | Use pause to create an animation. (1C)  **NOTE**: We don’t care about the pattern of the animation. It can be as simple as just putting a pause after the end of the inner loop, which creates a row by row animation, or any other more interesting ways.  The length of the animation should not be too long (i.e. no more than 20 seconds.). (1S) | 1 | 1 |
| GENERAL |  |  | Total=10 |
| G1 | Script starts with a concise comment describing the program.  Function comment follows function header. |  | 1 |
| G2 | Code is sufficiently (but not excessively) commented. |  | 1 |
| G3 | Line lengths are not excessively long (80 columns).  **NOTE**: It's ok if a couple lines are a little too long, especially if it’s due to having to print a very long string**.** |  | 1 |
| G4 | No extra output (debugging output) produced |  | 1 |
| G5 | Proper indentation is always used. |  | 1 |
| G6 | Use meaningful variable names. Do not overwrite MATLAB keywords. |  | 1 |
| G7 | Name important parameters as variables (constants). |  | 1 |
| G8 | No superfluous code (e.g., an empty if or else branch or a useless loop). Of course some students will have code that is awkward or unclear or inefficient. This point is specifically for not having code that does literally nothing. |  | 1 |
| G9 | Reasonably efficient code. |  | 1 |
| G10 | Does NOT put semicolon at wrong places, e.g., at the end of these lines: "if", "elseif", "else"," for","while", "function". |  | 1 |
| TOTAL |  | 22 | 19 |

**Penalties**

|  |  |  |
| --- | --- | --- |
| P1 | Student's code does not execute (or student provides a script when a function is required and vice-versa) | -1 from final score |
| P2 | Student's code crashes or does not terminate (infinite loop) for normal cases. | -1 from final score |
| P3 | All function headers and file names match those specified in the project description exactly. All input and output variables should be of the correct type. | -1 from final score |

**Grade Calculation**

|  |  |
| --- | --- |
| Total Possible Correctness Points | TC = 22 |
| Total Possible Style Points | TS = 19 |
| Student Correctness Points | C = min( \_\_\_ + 1 freebie point, TC) |
| Student Style Points | S = min( \_\_\_ + 1 freebie point, TS) |

Exceptions: If any file is missing/unacceptable, no freebie points can be applied to that file and subtract 3 style points for each missing/unacceptable file.

|  |  |
| --- | --- |
| Student’s final score | ([(C/TC)+(S/TS)] X 5) - Penalties  (Out of 10; 1 decimal; no negative score; round to NEAREST) |