**Project 4 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item no.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | isValidPartition.m | Total = 1 | Total = 2 |
| 1 | Returns 0 if the input does not contain all integers 1,..,9. | 1 |  |
| 2 | Does not crash if the input contains non-integer values. (Do not take off a penalty point if this causes a crash.) This is worth 0 points -- just warn them if this could cause a crash. | 0 |  |
| 3 | Functions correctly when the input is a 3x3 array (1S), and when the input is a length-9 row or column vector (1S). |  | 2 |
|  | isValidSudoku.m | Total = 4 | Total = 1 |
| 4 | Calls isValidPartition on each row of the input. | 1 |  |
| 5 | Calls isValidPartition on each column of the input. | 1 |  |
| 6 | Calls isValidPartition on each of the specified 3x3 subarrays. | 1 |  |
| 7 | Returns 1 if and only if the above checks succeed. | 1 |  |
| 8 | Uses a loop to perform the checks, rather than hard-coding many calls to isValidPartition. |  | 1 |
| PROBLEM 2 | springMass.m | Total = 2 | Total = 0 |
| 9 | The problem was attempted (no points deducted for an incorrect solution). | 2 |  |
| PROBLEM 3 | multiZoom.m | Total = 13 | Total = 6 |
| 10 | Correctly uses the jName parameter instead of hardcoding a file name. | 1 |  |
| 11 | Reads the image and creates the image matrix correctly (e.g. using imread()). | 1 |  |
| 12 | Shows the image correctly (using either imshow() or image()). | 1 |  |
| 13 | Uses the *title* command appropriately to give instructions to the user on the figure window. |  | 1 |
| 14 | Repeat the interaction with the user correctly using a while loop. | 1 |  |
| 15 | Gets input correctly by accepting 2 clicks. | 1 |  |
| 16 | Code works correctly when first set of clicks is a double click. |  | 1 |
| 17 | From 2nd set of clicks onwards, interaction ends if either width or height of selected "box" is <=4 (or <4) pixels. It’s OK if they instead check that the Euclidean distance or Manhattan distance is <=4, but make a note of the correct behavior. |  | 1 |
| 18 | Rounds the click coordinates to an integer value (1C). Does not crash when the user clicks outside the image (1C). (Do not take off a penalty point if it crashes in this case.) | 2 |  |
| 19 | The clicked area is correctly extracted as a subarray of the original image. | 2 |  |
| 20 | Interpolation is performed which results in type uint8 (1C). | 1 |  |
| 21 | Interpolation is correctly performed in the x (1C) and y (1C) directions, but not over the color dimension (1S). | 2 | 1 |
| 22 | Code is written to prevent uint8 overflow (e.g. by using *a/2+b/2* instead of *(a+b)/2*). |  | 1 |
| 23 | Correctly shows the new image (1C), and returns to the original image after a pause (1S). | 1 | 1 |
| GENERAL |  |  | Total = 10 |
| 24 | Script starts with a concise comment describing the program.  Function comment follows function header. |  | 1 |
| 25 | Code is sufficiently (but not excessively) commented. |  | 1 |
| 26 | Line lengths are not excessively long (80 columns).  NOTE: It's ok if a couple lines are a little too long, especially if they are print statements |  | 1 |
| 27 | No extra output (debugging output) produced |  | 1 |
| 28 | Proper indentation is always used. |  | 1 |
| 29 | Use meaningful variable names. Do not overwrite MATLAB keywords. **Note:** For this assignment, it is OK to overwrite the keyword “beta”. |  | 1 |
| 30 | Name important parameters as variables (constants). |  | 1 |
| 31 | 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 |
| 32 | Code runs in less than 1 minute on a lab computer. |  | 1 |
| 33 | Does NOT put semicolon at wrong places, e.g., at the end of these lines: "if", "elseif", "else"," for","while", "function". |  | 1 |
| TOTAL |  | 20 | 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 = 20 |
| 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) |