**Project 1 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item no.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | kepModel.m | Total = 9 | Total = 4 |
| 1 | Platonic solids are nested in the correct order: cube, tetrahedron, dodecahedron, icosahedron, octahedron. If one is missing, but the order is otherwise correct, then they still get the point. |  | 1 |
| 2 | Compute/initialize the radius of the outer sphere to 1. | 1 |  |
| 3 | Compute the edge length E of each platonic solid via: E=R/F\_R, where R is the radius of the enclosing sphere (previously computed) and F\_R is the proportionality constant relating edge length to outer-­radius. It is OK if this calculation appears implicitly in the calculation of the inner radius. | 1 |  |
| 4 | Compute the inner radius r of each platonic solid via: r=E \* F\_r, where E is the edge length of the platonic solid (previously  computed) and F\_r is the proportionality constant relating edge length and inner ­radius. It is ok if E is computed inline. | 1 |  |
| 5 | Proportionality constants F\_R and F\_r are computed correctly for each of the platonic solids. Subtract 1 point for each mistake, up to 3. | 3 |  |
| 6 | If the student doesn’t store the various proportionality constants F\_R and F\_r in  variables, remember to take off the general style point for assigning constants to variables (Item 33). |  |  |
| 7 | Compute the circumference of each sphere. | 1 |  |
| 8 | Values are computed for exactly six spheres. | 1 |  |
| 9 | Display both the radius and circumference of each sphere (1c). For this point, any display method is acceptable. Each value is displayed with 15 decimal places of precision (1s). | 1 | 1 |
| 10 | Values are displayed in a table, with one row per sphere and values lined up along the decimal point (1s). The table either has a table header (identifying the radius and circumference), or each value is labeled directly (1s). |  | 2 |
|  |  |  |  |
| PROBLEM 2 | almostSqr.m | Total = 14 | Total = 8 |
| 11 | Point 1 is set using user-clicked point (ginput). | 1 |  |
| 12 | Both variables x and y are set to a random number in the interval (1,9).  Uses built-in function rand() twice, once for each x and y. (1c)  Scales the result of rand() by 8. (1c)  Shifts the result of rand() by 1. (1c) | 3 |  |
| 13 | The code that solicits and plots Point 3 has been removed. |  | 1 |
| 14 | Points 1 and 2 are treated as the original opposing corners of a rectangle. | 1 |  |
| 15 | Correctly computes and assigns side lengths of the rectangle. | 1 |  |
| 16 | Determines which side length is long (L) or short (S) using *max()*/*min()* or an *if* statement. | 1 |  |
| 17 | Correctly uses if statements to check for each condition (L/S < 1.2; L/S is in the range of 1.2 to 2; L/S > 2). -1 for first mistake, -2 for more than one mistake. | 2 |  |
| 18 | Conditions in the *if* statement are simplified using an *if-elseif-else* construct. |  | 1 |
| 19 | In cases 2 & 3: Display the original ratio L/S (1s) and the average value between the height and width of the rectangle (1s) in the title area. |  | 2 |
| 20 | In cases 2 & 3: The new rectangle is centered at (5,5). | 1 |  |
| 21 | Draw 4 lines to form a rectangle (e.g. using *plot* or *rectangle*). |  | 1 |
| 22 | The figure window passes the “Sanity Check”: “The final figure window should show a rectangle, two markers that may or may not be the opposing vertices of the plotted rectangle, and a message in the title area.” |  | 1 |
|  | Handle all three conditions as follows: |  |  |
|  | Case 1: L/S is less than 1.2 |  |  |
| 23 | L/S is displayed to two  decimal places (1S) in the title area of the plot e.g. using title() and sprintf() (1C)  (Note: don’t take off for other conditions if display is not to two decimal places) | 1 | 1 |
|  | Case 2: L/S is in the range of 1.2 to 2 |  |  |
| 24 | Correctly performs one averaging step. | 1 |  |
|  | Case 3: L/S is greater than 2 |  |  |
| 25 | Correctly performs two averaging steps. -1 point for each error, max. -2. | 2 |  |
| 26 | Does not edit provided code/comments in sections where comments denote that the code should not be modified. |  | 1 |
| GENERAL |  |  | Total=10 |
| 27 | Script starts with a concise comment describing the program.  Function comment follows function header. |  | 1 |
| 28 | Code is sufficiently (but not excessively) commented. |  | 1 |
| 29 | 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 |
| 30 | No extra output (debugging output) produced |  | 1 |
| 31 | Proper indentation is always used. |  | 1 |
| 32 | Use meaningful variable names. Do not overwrite MATLAB keywords. |  | 1 |
| 33 | Name important parameters as variables (constants). |  | 1 |
| 34 | 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 |
| 35 | Reasonably efficient code. |  | 1 |
| 36 | Does NOT put semicolon at wrong places, e.g., at the end of these lines: "if", "elseif", "else"," for","while", "function". |  | 1 |
| TOTAL |  | 23 | 22 |

**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 = 23 |
| Total Possible Style Points | TS = 22 |
| 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) |