**Project 2 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item no.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | verify.m | Total = 6 | Total = 3 |
| 1 | The values of the inequality are computed correctly. (-1 for each error; max -2). | 2 |  |
| 2 | The program displays the correct message “the inequalities hold for all n values”. | 1 |  |
| 3 | Uses a conditional inside the loop to correctly check whether the inequality holds for a given n. | 1 |  |
| 4 | Uses a conditional outside the loop to correctly check whether the inequality holds for all n. | 1 |  |
| 5 | Prints 10 rows of values. | 1 |  |
| 6 | Values are printed in a table format (1S) and are labeled (1S). |  | 2 |
| 7 | Conditionals are fully simplified (i.e. they contain no extraneous clauses). Do not also deduct from #34 for this. |  | 1 |
| PROBLEM 2 | robotArm.m | Total = 9 | Total = 4 |
| REMINDER | Code should work for all r1,r2>0 and 0 ≤ alpha ≤ pi/2. Note that D(0) ≥ 0 for 0 ≤ alpha ≤ pi/2, so they are allowed to assume that D(0) is positive. |  |  |
| 8 | Correctly calculates the value D(beta). | 1 |  |
| 9 | Starts checking for solutions at beta=0. | 1 |  |
| 10 | Repeatedly increments beta by some positive value dBeta (1C). The value dBeta is reasonable (dBeta <= 0.3) (1C). | 2 |  |
| 11 | Uses a while loop which terminates when a sign change is detected. (-1 for a minor error in the loop condition, -2 max.) | 2 |  |
| 12 | Plots D(beta) against beta for each intermediate beta (1C). The plot has a title which displays the values of r1, r2, and alpha (1S). | 1 | 1 |
| 13 | Correctly handles the case when alpha = 0 or alpha = pi/2. In these cases, there is just a single point on the graph. | 1 |  |
| 14 | All parameters are set to their original values (r1=4, r2=3, alpha=pi/6). |  | 1 |
| 15 | Prints the final value of beta to the Command Window (1C). The value should be labeled (1S). | 1 | 1 |
| 16 | No other values of beta are printed to the Command Window. |  | 1 |
| PROBLEM 3 | printDiamond.m | Total = 4 | Total = 2 |
| 17 | The program prints out 2n-1 lines for any input n>3. | 1 |  |
| 18 | The first and last lines have exactly 1 asterisk, and the other lines have exactly 2 asterisks. | 1 |  |
| 19 | The program outputs a diamond exactly matching the specified figure. (-1 if the output looks like a diamond but does not match the figure; -2 max.) | 2 |  |
| 20 | The program is well organized using nested loops. |  | 1 |
| 21 | The program does not contain any unstructured control statements (e.g. “break”). |  | 1 |
| GENERAL |  |  | Total = 10 |
| 22 | Script starts with a concise comment describing the program.  Function comment follows function header. |  | 1 |
| 23 | Code is sufficiently (but not excessively) commented. |  | 1 |
| 24 | 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 |
| 25 | No extra output (debugging output) produced |  | 1 |
| 26 | Proper indentation is always used. |  | 1 |
| 27 | Use meaningful variable names. Do not overwrite MATLAB keywords. **Note:** For this assignment, it is OK to overwrite the keyword “beta”. |  | 1 |
| 28 | Name important parameters as variables (constants). |  | 1 |
| 29 | 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 |
| 30 | Reasonably efficient code. |  | 1 |
| 31 | Does NOT put semicolon at wrong places, e.g., at the end of these lines: "if", "elseif", "else"," for","while", "function". |  | 1 |
| TOTAL |  | 19 | 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 = 19 |
| 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) |