**Project 5 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item no.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | possibleGenes.m | Total = 14 | Total = 4 |
| 1 | Has the correct function definition:  *function n = findGenes(inFileName, outFileName)* (1C). Parameter names are meaningful and explained in the function’s comment (1S). | 1 | 1 |
| 2 | Opens the input file using *fopen*. If the student appends “.txt” to the file name, leave a note discouraging this, but do not take off points. | 1 |  |
| 3 | Uses a loop or other method to load data from the input file. | 1 |  |
| 4 | Skips the first (header) line of input. | 1 |  |
| 5 | Joins together all subsequent lines of input. (Genes may span multiple lines.) | 1 |  |
| 6 | Closes all opened files with *fclose*. |  | 1 |
| 7 | Searches for genes in all three possible reading frames (-1 if student attempted to search the reading frames but made a mistake, -2 max). | 2 |  |
| 8 | Checks for the start codon “ATG”. | 1 |  |
| 9 | Correctly handles two consecutive start codons. (No gene is recorded, and the start index is updated to the last start codon.) | 1 |  |
| 10 | Correctly handles an empty gene. (No gene is recorded.) | 1 |  |
| 11 | Searches for the stop codon “TAG”/”TAA”/”TGA”. | 1 |  |
| 12 | Decomposes the problem using at least one subfunction OR by organizing code into clear logical blocks. |  | 1 |
| 13 | Returns the number of genes found. | 1 |  |
| 14 | Writes to the file *outFileName*. | 1 |  |
| 15 | The output file contains one line per gene found (1C). Each line contains the gene with its start and stop codon removed (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 they are print statements |  | 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. **Note:** For this assignment, it is OK to overwrite the keyword “beta”. |  | 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 |  | 14 | 14 |

**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 = 14 |
| Total Possible Style Points | TS = 14 |
| 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) |