**Project 6 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item #.** | **Description** | **Correctness** | **Style** |
| PROBLEM A | Room.m | Total = 29 | Total = 6 |
| 1 | Room constructor:  -Initializes a room object with fields set to the 7 actual parameters only if 7 arguments are given (1C).  -Else, sets the 7 properties to be reasonable default values, e.g. -1, 0, etc. (1S).  -Also sets playerVisited to 0 (1C). | 2 | 1 |
| 2 | getLoc() returns x- and y-coordinates of current room | 1 |  |
| 3 | getID() returns room's private ID number | 1 |  |
| 4 | isHazardous() returns 1 if room is hazardous, 0 otherwise | 1 |  |
| 5 | applyHazard():  -Calls decreaseHealth if hazardID = 1 (1C),  -and poisons the player if hazardID=2 (1C).  -Uses the hazardAmount constant instead of a hardcoded value, e.g. 20 (1S) | 2 | 1 |
|  | Character.m |  |  |
| 6 | Character constructor:  -Initalizes a character object with the room property set to startRoom if nargin == 1 (1C).  -If nargin ~= 1, sets the room property to be a default Room object using Room() (1S). | 1 | 1 |
| 7 | moveCharacter() method:  -Creates new x- and y-coordinates by adding dx and dy to self.room.xCoord and self.room.yCoord, respectively (1C).  -Gets new room by referencing an existing room in roomArr, e.g. via roomArr(newX,newY) or equivalent (1C).  **NOTE**: Do NOT take points off if the students don’t adjust self.room in moveCharacter but correctly set it in Player’s and Monster’s methods. | 2 |  |
|  | Player.m |  |  |
| 8 | Player constructor:  -Calls superclass constructor via @Character(startRoom) (1C).  -Sets player's health, poisonHit, and poisonEscape properties to be the given arguments and also sets playerInRoom property of startRoom to be 1 (1C).  -Sets health, poisonHit, and poisonEscape to be reasonable default values if 4 arguments not given (1S). | 2 | 1 |
| 9 | getHealth() returns value of player's health property (1 point). | 1 |  |
| 10 | decreaseHealth() sets player's health to be health-damage (1 point) | 1 |  |
| 11 | checkPoison() returns value of player's poisoned property (1 point) | 1 |  |
| 12 | move() method:  -Only proceeds if health > 0 (1C).  -Calls superclass' moveCharacter() method with correct arguments, and also sets playerInRoom and playerVisited to both be 1 (1C).  -Poison handling (2C):  -(a) checking if player is poisoned and decreasing health if so,  -(b) determining whether player can escape from poison using a random number,  -(c) checking if new room is hazardous and applying hazards if so;  \*award 1 out of the 2 points if any of (a), (b), (c) is missing. | 4 |  |
|  | Monster.m |  |  |
| 13 | Monster constructor:  -Sets monsterInRoom property of startRoom to 1 (1C)  -Calls superclass constructor via @Character(startRoom) (1C). | 2 |  |
| 14 | moveToAttack() method:  -Correctly finds minimizing point in x & y direction (2C):  \*Can be done with two for loop method or with comparing the coordinates  -Calls moveCharacter correctly (1C)  -Updates monsterInRoom (1C) | 4 |  |
| 15 | moveToProtect() method (same as above):  -Correctly finds minimizing point in x & y direction (2C):  \*Can be done with two for loop method or with comparing the coordinates  \*ALSO, two values are correct here, the point minimizing the average distance between the exit and the player, and the exit itself as said in solutions  -Calls moveCharacter correctly (1C)  -Updates monsterInRoom (1C)  **NOTE**: If the students only minimize the distance between the monster and the exit, points should not be taken off as it’s specified in the rubric. | 4 |  |
|  | General Style |  |  |
| 16 | Does not change the draw() or removeDrawing() methods in the Room, Player and Monster classes. Does not change/add any properties in any class or change the accessibility of any properties/methods in any class. Does not modify Game class at all. (2S, all or nothing) |  | 2 |
| PROBLEM 2 | RandomMondrian.m | Total = 7 | Total =  3 |
| 17 | Use variables (pHalt, MyColors and bWidth) to represent the parameters and initialize them at the beginning of the function. |  | 1 |
| 18 | The termination condition of recursion is level == 0 || rand < pHalt. (2C, 1C for each of the conditions) | 2 |  |
| 19 | Pick a random color in the color map. |  | 1 |
| 20 | Use fill to fill the rectangle with the random color. | 1 |  |
| 21 | Use plot to draw the border lines. | 1 |  |
| 22 | Randomly pick a partitioning point in the rectangle (1C). When generating the random point, there should be some constraint that the point is not too near the edge of the input rectangle. (1S) | 1 | 1 |
| 23 | Recursively call RandomMondrian to partition the four small rectangles using correct input arguments. (2C, -1C for each mistake) | 2 |  |
| 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 |  | 36 | 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 = 36 |
| 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) |