Unit Testing Plan/Result for void SetCurrentValue(int) and GetCurrentValue()

| Test Description | Setup | Expected Result | Actual Result |
| --- | --- | --- | --- |
| Int Value is valid | - Create currentValue member variable  - Call SetCurrentValue with an int argument between 1-9 inclusive  - Call and print GetCurrentValue | - value is assigned to currentValue  - currentValue printed with same value passed through parameter | - value is assigned to currentValue  - currentValue printed with same value passed through parameter |
| Int value is not valid | -- Create currentValue member variable  - Call SetCurrentValue with an int argument less than or equal to 0 or greater than 9  - Call and print GetCurrentValue | - value is not assigned to currentValue  - currentValue printed as 0 (default value) | - value is not assigned to currentValue  - currentValue printed as 0 (default value) |

Unit Testing Plan/Result for GameEngine Constructor (similar to Setup function)

| Test Description | Setup | Expected Result | Actual Result |
| --- | --- | --- | --- |
| All member variables/objects created and initialized without loading puzzle | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of Cell objects)  - Create a history stack class (no functionality needed for test) | - No crash  - Puzzle and history pointers stored in GameEngine object (successfully linked - no memory issues or segmentation faults)  - Other member variables initialized to their respective default values | - No crash  - No memory issues/segmentation faults  - Other member variables initialized to their respective default values |
| All member variable/objects created and initialized after loading puzzle in main | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of Cell objects)  - Create a history stack class (no functionality needed for test)  - Puzzle set from array of values called “EasyPuzzle” in main  - Print puzzle function in GameEngine class to visually see if puzzle is linked to GameEngine object | - No crash  - No memory issues or segmentation faults  - Puzzle printed in main and from GameEngine class and are the same puzzle  - Other member variables initialized to their respective default values | - No crash  - No memory issues or segmentation faults  - Puzzle printed in main and from GameEngine class and are the same puzzle  - Other member variables initialized to their respective default values |

Unit Testing Plan/Result for SetValue(int, int)

| Test Description | Setup | Expected Result | Actual Result |
| --- | --- | --- | --- |
| row / col not valid | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of int objects)  - Call SetCell with invalid arguments | -Nothing happens  - Function exits | -Nothing happens  - Function exits |
| currentValue wasn’t previously set to a valid number | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of int objects)  - Call SetCell with valid arguments | -Nothing happens  - Function exits | -Nothing happens  - Function exits |
| Load puzzle and set valid cell | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of Cell objects)  - create a simple Cell class from a minimal implementation (SetRow, SetCol, SetValue, int row, int col, int val)  - load puzzle in main  - load GameEngine constructor  - Call SetCurrentValue with valid argument  - Call SetCell with valid arguments | - print puzzle before with no updated cell value  - print puzzle after with cell at (row, col) having the value passed into SetCurrentValue  - no crash  - No memory issues | - print puzzle before with no updated cell value  - print puzzle after with cell at (row, col) having the value passed into SetCurrentValue |
| Load puzzle and set valid cell, then push onto history stack | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of Cell objects)  - create a simple Cell class from a minimal implementation (SetRow, SetCol, SetValue, int row, int col, int val)  - create a simple History class from a minimal implementation (pushHistory(int val), checkForEmpty(), popHistory())  - load puzzle in main  - load GameEngine constructor  - Call SetCurrentValue with valid argument  - Call SetCell with valid arguments  - push currentValue onto history stack (to keep it simple)  - call above function^ twice to ensure stack works properly | -print puzzle before with no updated cell value  - print puzzle after with cell at (row, col) having the value passed into SetCurrentValue  - no crash  - No memory issues  - print value from PopHistory() shows second value pushed | -print puzzle before with no updated cell value  - print puzzle after with cell at (row, col) having the value passed into SetCurrentValue  - no crash  - No memory issues  - print value from PopHistory() shows second value pushed |

Unit Testing Plan/Result for void SetNote()

| Test Description | Setup | Expected Result | Actual Result |
| --- | --- | --- | --- |
| row / col are invalid values | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of int objects)  - Create a simple Cell class (setNotes(), notes array of ints)  - Call SetNote with invalid arguments | -Nothing happens  - Function exits | -Nothing happens  - Function exits |
| Row and col are valid values | - Create a puzzle from a minimal implementation of puzzle class ( with SetCell() and GetCell(), 2D array of int objects)  - create simple Cell class (SetNotes(), notes array of ints)  - Call SetNote with valid arguments | - Print notes before  - print notes after with updated notes change and in correct position in array  - no crash  - return notes array | - Print notes before  - print notes after with updated notes change and in correct position in array  - no crash  - return notes array |