Name: _____

Rubric for Assignment #2

Criteria		Unsatisfactory	Acceptable	Good	Exceptional	Marks
		0	1	2	3	
Functionality	**Stack/Queue**	- did not use a stack or queue to solve the maze	- a stack and/or a queue was used, but it was not used to solve the maze - recursion was used - many small or a few large errors exist	- a stack and/or a queue was used to solve the maze - recursion was not used - a few small errors exist	- a custom built stack and/or a queue was designed and used to solve the maze - no recursion was used and no errors exist in the implementation of the stack/queue	x 4 _____
	Maze	- did not find the exit of the maze	- the maze exit was found and a path from the entrance to the exit exists - a single path was not found - includes paths to dead-ends - many small or a few large errors exist	- a single path to the exit was found - no dead-ends are shown - a few small errors in the solution or the display exist	- the single solution to from the entrance to the exit of the maze was found and displayed using a special character. - the solution file for the example maze, generated by the program, matches the example solution	x 4 _____
					Sub-Total	_____ **24**

Note : 24 x 25% = 6

Note: The following aspects of the program will only be graded if you receive more than 25% on the functionality rubric.

Output	**Aesthetics**	- incorrect or not existent use of whitespace in output - output is confusing and hard to follow	- fair use of whitespace - most output is clear, but poorly presented	- good use of whitespace - output is clear and fairly well presented	- excellent use of whitespace - output is clear and attractively presented	_____
Source Code	**Readability**	- source code is poorly organized and very difficult to read	- source code can be read, but is hard to follow	- source code is fairly easy to read, but is hard to follow in some areas	- source code is exceptionally well organized and easy to follow	_____
	Reusability	- source code cannot be reused - no functions or classes used	- small sections of code could be reused	- large portions of code could be reused with some modifications	- source code could be easily reused with little modifications	_____
	Efficiency	- contains large portions that could have been easily reduced using a different method - a lot of code is duplicated, copy/pasted	- tried some methods to improve efficiency - can explain what they attempted	- employed good ideas to improve efficiency - can point out where other improvements could be made	- very clean and efficient code - can propose new ideas for improvement	_____
	Comments	- little to no comments used	- comments are used, some are meaningful and easily understood - some files and functions have headers	- comments are used extensively, most are meaningful and easily understood - most files/functions have headers	- not over/under commented - comments are meaningful and easily understood - files/functions have headers - is self-documenting	_____
	Naming Convention	- no standard naming convention followed	- a standard naming convention was used for part of the program, but deviated often	- a standard naming convention was used for most of the program and deviated very little	- industry standard naming convention used throughout the program	_____
	Consistency	- no consistency in formatting or layout of source code	- source code formatting was fairly consistent, but contained some inconsistency with whitespace, brackets, etc	- source code formatting was very consistent with respect to whitespace, brace brackets, parentheses, etc	- source code formatting never deviated from the programmer's layout	_____

Assignment: _____

Sub-Total	_____ **21**
Total	_____ **45**