DE LA SALLE UNIVERSITY

2401 Taft Avenue, Manila 1004, Philippines

College of Computer Science
Department of Software Technology

Programming with Structured Data Types (CCPROG2)

Project Title : Minesweeper
Project Proponent/s : Mary Eunice E. Griño & Mariel M. Tamondong
Project Description : In this project, students must implement a C program for the game Minesweeper. The program should allow a user to play the Minesweeper game multiple times, following the rules of the game which are outlined below. Additionally, the program must allow the user to create their own Minesweeper levels which can be saved in text files and retrieved later.

Test Script

Function Name	#	Test Description	Test Input	Expected Result	Actual Result	P/F
delay	1	Short time delay of program execution	delay(100)	Program execution is delayed by 100 milliseconds	Program execution is delayed by 100 milliseconds	P
	2	Short time delay of program execution	delay(1000)	Program execution is delayed by 1 second	Program execution is delayed by 1 second	P
	3	Short time delay of program execution	delay(5000)	Program execution is delayed by 5 seconds	Program execution is delayed by 5 seconds	P
controlsMenu	1	Input is an up or down arrow key and the selection is within bounds	max == 4 selection == 2 The player presses on the down arrow	selection = 3, returns 3	selection = 3, returns 3	P

			key and enter			
	1	Input is an arrow key, but not an up or down arrow key	The player presses down on the left arrow key	It prints a message telling the player to only use up and down arrow keys.	It prints a message telling the player to only use up and down arrow keys.	P
		Input is not an arrow key and not enter	The player presses down on a non-arrow key, except for enter	It prints a message telling the player to only use arrow keys.	It prints a message telling the player to only use arrow keys.	P
		The player makes the selection go out of bounds	max == 4 selection == 3 The player presses on the down arrow key	Nothing will happen, the selection will remain at 3.	Nothing will happen, the selection will remain at 3.	P
controlsGame	1	Input is an arrow key and the selection is still within bounds	level.rows == 5 level.cols == 5 row == 3 col == 3 The player presses on the right arrow key then enter	row = 3 col = 4 It will print the gameBoard and the arrow showing the position of row 3 and column 4. Then, the value of *rowChosen and *colChosen will be changed to the values of row and col, respectively.	row = 3 col = 4 It will print the gameBoard and the arrow showing the position of row 3 and column 4. Then, the value of *rowChosen and *colChosen will be changed to the values of row and col, respectively.	P
		Input is not an arrow key and not enter	The player presses down on a non-arrow	It prints a message telling the player to	It prints a message telling the player to	P

			key, except for enter	only use arrow keys.	only use arrow keys.	
		The player makes the selection go out of bounds	level.rows == 5 level.cols == 5 row == 3 col == 5 The player presses on the right arrow key	Nothing will happen, the selection will remain at row 3 column 5.	Nothing will happen, the selection will remain at row 3 column 5.	P
controlsLevelEdit	1	Input is an arrow key and the selection is still within bounds	level.rows == 5 level.cols == 5 row == 2 col == 2 The player presses on the up arrow key then enter.	row = 1 col = 2 It will print the board and the arrow showing the position of row 1 and column 2. Then, the value of *rowChosen and *colChosen will be changed to the values of row and col, respectively.	row = 1 col = 2 It will print the board and the arrow showing the position of row 1 and column 2. Then, the value of *rowChosen and *colChosen will be changed to the values of row and col, respectively.	P
		Input is not an arrow key and not enter	The player presses down on a non-arrow key, except for enter	It prints a message telling the player to only use arrow keys.	It prints a message telling the player to only use arrow keys.	P
		The player makes the selection go out of bounds	level.rows == 10 level.cols == 9 row == 0 col == 0 The player presses	Nothing will happen, the selection will remain at row 0 column 50.	Nothing will happen, the selection will remain at row 0 column 0.	P

			on the left arrow key			
makeBoard	1	Create a board with 1 mine.	level->rows == 5 level->cols == 5 level->mines == 1	A 5x5 board will be initialized with '.' and one randomly chosen tile will be initialized with 'X'	A 5x5 board will be initialized with '.' and one randomly chosen tile will be initialized with 'X'	P
	2	Create a board with all tiles as mines.	level->rows == 5 level->cols == 5 level->mines == 25	A 5x5 board will be initialized with all values being 'X'	A 5x5 board will be initialized with all values being 'X'	P
	3	Create a board with no mines.	level->rows == 5 level->cols == 5 level->mines == 0	A 5x5 board will be initialized with all values being '.'	A 5x5 board will be initialized with all values being '.'	P
mineCount	1	The given indices have no surrounding mines on the board	i = 5 j = 5 board[5][5] == '.' board[4][5] == '.' board[6][5] == '.' board[5][4] == '.' board[5][6] == '.' board[4][4] == '.' board[4][6] == '.' board[6][4] == '.' board[6][6] == '.'	Return: 0	Return 0:	P
	2	The given indices have surrounding mines on the board	i = 5 j = 5 board[5][5] == '.' board[4][5] == 'X' board[6][5] == '.' board[5][4] == '.'	Return: 2	Return: 2	P

			board[5][6] == '.' board[4][4] == 'X' board[4][6] == '.' board[6][4] == '.' board[6][6] == '.'			
	3	The given indices is a mine on the board	i = 7 j = 4 board[7][4] == 'X'	Return: -1	Return: -1	P
cascade	1	The given indices are out of bounds (Less than 0 or greater than or equal to the level dimensions).	level->rows == 6 level->cols == 6 i == 6 j == 3	Returns	Returns	P
	2	The given indices have a mineCount that is greater than 0.	level->gameBoard[i][j] == 1	Returns	Returns	P
	3	The given indices have a mineCount that is equal to 0.	level->gameBoard[i][j] == 0	It recurses with indices that surround the original, and stops when the revealed tiles are greater than 0.	It recurses with indices that surround the original, and stops when the revealed tiles are greater than 0.	P
placeFlag	1	The player chooses to flag a tile that is hidden.	level->gameBoard[5][5] == HIDDEN i == 5 j == 5	level->gameBoard[5][5] == FLAG	level->gameBoard[5][5] == FLAG	P
	2	The player chooses to flag a tile that is already flagged.	level->gameBoard[5][5] == FLAG	It will print a message telling the player that the tile is already	It will print a message telling the player that the tile is already	P

			i == 5 j == 5	flagged.	flagged.	
	3	The player chooses to flag a tile that is already revealed.	level->gameBoard[5][5] == 0 i == 5 j == 5	It will print a message telling the player that the tile is already revealed.	It will print a message telling the player that the tile is already revealed.	P
removeFlag	1	The player chooses to remove a flag on a tile that is flagged.	level->gameBoard[5][5] == FLAG i == 5 j == 5	level->gameBoard[5][5] == HIDDEN	level->gameBoard[5][5] == HIDDEN	P
	2	The player chooses to remove a flag on a tile that is not flagged.	level->gameBoard[5][5] != FLAG i == 5 j == 5	It will print a message telling the player that the tile is not flagged.	It will print a message telling the player that the tile is not flagged.	P
	3	The player chooses to remove a flag on an edge tile.	level->gameBoard[0][0] == FLAG i == 0 j == 0	level->gameBoard[0][0] == HIDDEN	level->gameBoard[0][0] == HIDDEN	P
inspectBoard	1	The player inspects a tile that contains a mine.	level->board[6][7] == ‘X’ i = 6 j = 7	level->gameBoard[i][j] == 999 Return: 0	level->gameBoard[i][j] == 999 Return: 0	P
	2	The player inspects a tile that does not contain a mine.	level->board[8][8] == ‘.’	It will run the cascade function.	It will run the cascade function.	P

			i == 8 j == 8	Return: 1	Return: 1	
	3	The player inspects a tile that is flagged.	level->gameBoard[9][10] == FLAG i == 9 j == 10	It will print a message telling the user that they can not inspect a flagged tile. Return: 1	It will print a message telling the user that they can not inspect a flagged tile. Return: 1	P
gameChecker	1	The revealed tiles is equal to the non-mine tiles in the board.	level.rows == 6 level.cols == 6 Level.mines == 2 revealedCount == 34	It will print a message telling the user that they won and the outcome of the game will be set to “win.” Return: 0	It will print a message telling the user that they won and the outcome of the game will be set to “win.” Return: 0	P
	2	The revealed tiles is less than the non-mine tiles in the board.	level.rows == 6 level.cols == 6 Level.mines == 2 revealedCount == 30	Return: 1	Return: 1	P
	3	The revealed tiles is 0.	level.rows == 6 level.cols == 6 Level.mines == 2 revealedCount == 0	Return: 1	Return: 1	P
fileExists	1	The given filename exists	strcmp(filename, “profiles/gem.txt”) == 0	The function will open *file with filename in read mode then upon successful read, it will close the file.	The function will open filename in read mode then upon successful read, it will close the file.	P

				Return: 1	Return: 1	
	2	The given filename does not exist	strcmp(filename, “profiles/gem.txt”) == 0	The function will open *file with filename in read mode, but will not find file. Return: 0	The function will open *file with filename in read mode, but will not find file. Return: 0	P
	3	The given filename is not a .txt file.	strcmp(filename, “profiles/gem.c”) == 0	The function will open *file with filename in read mode, but will not find file. Return: 0	The function will open *file with filename in read mode, but will not find file. Return: 0	P
transferSnapshot	1	The destFile[] and sourceFile[] contain previously played games.	destFile[] == recentgame1.txt sourceFile[] == recentgame0.txt	recentgame1.txt will be overwritten with the values in recentgame0.txt	recentgame1.txt will be overwritten with the values in recentgame0.txt	P
	2	The sourceFile[] does not contain a played game (i.e. 0).	destFile[] == recentgame1.txt sourceFile[] == recentgame0.txt	recentgame1.txt will have been printed with 0.	recentgame1.txt will have been printed with 0.	P
	3	The sourceFile[] does not exist.	destFile[] == recentgame1.txt sourceFile[] == recentgame.txt	The function will end without reading and writing the sourceFile[] and destFile, respectively.	The function will end without reading and writing the sourceFile[] and destFile, respectively.	P
saveSnapshot	1	The outcome of the game is “lose”	strcmp(outcome, “lose”) == 0	The most recent game of the current user will	The most recent game of the current user will	P

				be overwritten with the current game. The snapshot will include the revealed tiles, hidden tiles, mines, mine that exploded, and the flags.	be overwritten with the current game. The snapshot will include the revealed tiles, hidden tiles, mines, mine that exploded, and the flags.	
	2	The outcome of the game is “win”	strcmp(outcome, “win”) == 0	The most recent game of the current user will be overwritten with the current game. The snapshot will include the revealed tiles and mines.	The most recent game of the current user will be overwritten with the current game. The snapshot will include the revealed tiles and mines.	P
	3	The outcome of the game is “quit”	strcmp(outcome, “quit”) == 0	The most recent game of the current user will be overwritten with the current game. The snapshot will include the revealed tiles, hidden tiles, and flags placed.	The most recent game of the current user will be overwritten with the current game. The snapshot will include the revealed tiles, hidden tiles, and flags placed.	P
updateStatistics	1	The outcome of the custom game is “lose”	strcmp(outcome, “lose”) == 0 strcmp(level.mode, “CUSTOM”) == 0	currentUser->games_lost_custom++ The text file of the currentUser will then be rewritten with the updated statistics.	currentUser->games_lost_custom++ The text file of the currentUser will then be rewritten with the updated statistics.	P
	2	The outcome of a classic game is “win”	strcmp(outcome, “win”) == 0	currentUser->games_won_classic++	currentUser->games_won_classic++	P

			strcmp(level.mode, “CLASSIC-EASY”) == 0	The text file of the currentUser will then be rewritten with the updated statistics.	The text file of the currentUser will then be rewritten with the updated statistics.	
	3	The outcome of a custom game is “quit”	strcmp(outcome, “quit”) == 0 strcmp(level.mode, “CUSTOM”) == 0	currentUser->games_lost_custom++ The text file of the currentUser will then be rewritten with the updated statistics.	currentUser->games_lost_custom++ The text file of the currentUser will then be rewritten with the updated statistics.	P
gameProper	1	The player chooses to inspect the board, but the selected tile is a mine.	choice == 0	The inspectBoard function is called. alive == 0 saveSnapshot and updateStatistics will then be called.	The inspectBoard function is called. alive == 0 saveSnapshot and updateStatistics will then be called.	P
	2	The player chooses to inspect the board, and the selected tile is not a mine.	choice == 0	The inspectBoard function is called. alive == 1 Then, checks if all the non-mine tiles are revealed. If yes, alive == 0 and the game ends. saveSnapshot and updateStatistics will then be called.	The inspectBoard function is called. alive == 1 Then, checks if all the non-mine tiles are revealed. If yes, alive == 0 and the game ends. saveSnapshot and updateStatistics will then be called.	P

	3	The player chooses to place a flag.	choice == 1	The placeFlag function is called.	The placeFlag function is called.	P
	4	The player chooses to remove a flag.	choice == 2	The removeFlag function is called.	The removeFlag function is called.	P
	5	The player chooses to quit.	choice == 3	alive == 0 saveSnapshot and updateStatistics will then be called.	alive == 0 saveSnapshot and updateStatistics will then be called.	P
checkLevels	1	There are no existing custom levels.	First line in the LVL_DIR is 0	It will not print any levels. Return: 0	It will not print any levels. Return: 0	P
	2	There are existing custom levels.	First line in the LVL_DIR is 5 and there are 5 levels.	It will print the names of the 5 levels. Return: 5	It will print the names of the 5 levels. Return: 5	P
	3	There is exactly one custom level.	First line in the LVL_DIR is 1 and there is 1 level.	It will print the name of the 1 level. Return: 1	It will print the name of the 1 level. Return: 1	P
placeMine	1	The given indices do not contain a mine on the board.	customLevel->board[5][5] == '.' row == 5 col == 5	customLevel->board[5][5] == 'X' It will print a message saying that the mine is placed.	customLevel->board[5][5] == 'X' It will print a message saying that the mine is placed.	P
	2	The given indices already contain a mine on the	customLevel->board[5][5] == 'X'	customLevel->board[5][5] == 'X'	customLevel->board[5][5] == 'X'	P

		board.	row == 5 col == 5	It will print a message saying that position is invalid.	It will print a message saying that position is invalid.	
	3	The given indices are out of bounds.	customLevel->rows == 5 customLevel->cols == 5 row == 5 col == 4	It will print a message saying that position is invalid.	It will print a message saying that position is invalid.	P
deleteMine	1	The given indices do not contain a mine on the board.	customLevel->board[5][5] == '.' row == 5 col == 5	It will print a message saying that there is no mine.	It will print a message saying that there is no mine.	P
	2	The given indices contain a mine on the board.	customLevel->board[5][5] == 'X' row == 5 col == 5	customLevel->board[5][5] == '.' It will print a message saying that the mine has been removed.	customLevel->board[5][5] == '.' It will print a message saying that the mine has been removed.	P
	3	The given indices are out of bounds.	customLevel->rows == 5 customLevel->cols == 5 row == 5 col == 4	It will print a message saying that position is invalid.	It will print a message saying that position is invalid.	P
checkValidity	1	Created level has some mines placed	customLevel->rows == 5	cells != minesCount && minesCount < cells	cells != minesCount && minesCount < cells	P

			customLevel-cols == 5 minesCount == 2	Return: 1	Return: 1	
	2	Created level has no mines placed	customLevel->rows == 4 customLevel-cols == 4 minesCount == 16	cells == minesCount The function will print “Invalid level: Every cell contains a mine.” Return: 0	cells == minesCount The function will print “Invalid level: Every cell contains a mine.” Return: 0	P
	3	Created level has mines in every cell	customLevel->rows == 5 customLevel-cols == 5 minesCount == 0	minesCount == 0 The function will print “Invalid level: There are no mines placed.” Return: 0	minesCount == 0 The function will print “Invalid level: There are no mines placed.” Return: 0	P
deleteLevel	1	The given level exists.	strcmp(path, “levels/game.txt”) == 0	The function will delete the specified level file and update the level list, removing the file’s name.	The function will delete the specified level file and update the level list, removing the file’s name.	P
	2	The given level does not exist.	strcmp(path, “levels/test.txt”) == 0	The function will print “Level does not exist. Try again.” and prompt user back to level editor menu.	The function will print “Level does not exist. Try again.” and prompt user back to level editor menu.	P
	3	-	-	-	-	-
saveFile	1	Saving a new level file	strcmp(path,	The function will create	The function will create	P

			“levels/new.txt”) == 0	a new file and save the created level’s data, then update the level list accordingly.	a new file and save the created level’s data, then update the level list accordingly.	
	2	Updating an existing level file	strcmp(path, “levels/gem.txt”) == 0	The function will open existing file and overwrite the edited level’s data, then update the level list accordingly.	The function will open existing file and overwrite the edited level’s data, then update the level list accordingly.	P
	3	The file to be saved has the minimum values for a level.	customLevel->rows == 5 customLevel-cols == 5 minesCount == 1	The file is saved with the inputted parameters, and the level list will be updated accordingly.	The file is saved with the inputted parameters, and the level list will be updated accordingly.	P
editLevel	1	User attempts to save an invalid level	customLevel->rows == 5 customLevel-cols == 5 minesCount == 25	The function will call checkValidity() update the loop variables accordingly. save == 0 Level is not saved.	The function will call checkValidity() update the loop variables accordingly. save == 0 Level is not saved	P
	2	User chooses to go back to menu	choice == 3	The function will print “You have opted to go back.” quit == 0 Loop breaks and returns to menu	The function will print “You have opted to go back.” quit == 0 Loop breaks and returns to menu	P

	3	User chooses to place mine	choice == 0	The function will call placeMine() allowing the player to place mines	The function will call placeMine() allowing the player to place mines	P
loadLevel	1	Loading an existing level	path[] == levels/gem.txt	The function will the specified file using path, allow the user to edit the level calling editLevel(), then save if they made valid changes	The function will the specified file using path, allow the user to edit the level calling editLevel(), then save if they made valid changes	P
	2	Loading a level that does not exist	path[] == levels/not.txt	The function will print “Level does not exist. Try again.” and have player return to menu	The function will print “Level does not exist. Try again.” and have player return to menu	P
	3	-	-	-	-	-
createLevel	1	Create a file with existing filename	filename[] == gem.txt	The function will print “Level cannot be created. File already exists.” and have player return to menu	The function will print “Level cannot be created. File already exists.” and have player return to menu	P
	2	Create a file that does not exist yet	filename[] == molly.txt customLevel->rows == 5 customLevel->cols == 10 mines == 5	The function will print “Level <provided filename> will be created.” Prompt user input for customLevel->rows and customLevel->cols, create the board.	The function will print “Level <provided filename> will be created.” Prompt user input for customLevel->rows and customLevel->cols, create the board.	P

				It then allows the user to edit the level further, printing “Level created successfully.” upon valid edits.	It then allows the user to edit the level further, printing “Level created successfully.” upon valid edits.	
	3	Create a file with invalid rows and columns input	filename[] == molly.txt customLevel->rows == 3 customLevel->cols == 4	The function will prompt “Invalid number of rows and columns.” until the user inputs valid values.	The function will prompt “Invalid number of rows and columns.” until the user inputs valid values.	P
	4	Create a file with name greater than 20 characters	strcmp(name, “12345678912345678 9123”) == 0	The function will create a level with only the first 20 characters of the input.	The function will create a level with only the first 20 characters of the input.	P
levelEditor	1	The player chooses to create a level.	choice == 0	The createLevel function will be called.	The createLevel function will be called.	P
	2	The player chooses to edit a level.	choice == 1	The loadLevel function will be called.	The loadLevel function will be called.	P
	3	The player chooses to delete a level.	choice == 2	The deleteLevel function will be called.	The deleteLevel function will be called.	P
	4	The player chooses to go back.	choice == 3	quit == 1, exiting the loop	quit == 1, exiting the loop	P
getStatistics	1	The current user has not played any games.	strcmp(currentUser.name, “USER”) == 0	currentUser->games_won_classic == 0	currentUser->games_won_classic == 0	P

				currentUser->games_lost_classic == 0 currentUser->games_won_custom == 0 currentUser->games_lost_custom == 0 currentUser->recentgame[0].path currentUser->recentgame[1].path currentUser->recentgame[2].path	currentUser->games_lost_classic == 0 currentUser->games_won_custom == 0 currentUser->games_lost_custom == 0 currentUser->recentgame[0].path currentUser->recentgame[1].path currentUser->recentgame[2].path	
	2	The current user has won and lost games, only in custom levels.	strcmp(currentUser.name, “USER”) == 0	currentUser->games_won_classic == 0 currentUser->games_lost_classic == 0 currentUser->games_won_custom == 3 currentUser->games_lost_custom == 2 currentUser->recentgame[0].path currentUser->recentgame[1].path	currentUser->games_won_classic == 0 currentUser->games_lost_classic == 0 currentUser->games_won_custom == 3 currentUser->games_lost_custom == 2 currentUser->recentgame[0].path currentUser->recentgame[1].path	P

				currentUser->recentgame[2].path	currentUser->recentgame[2].path	
	3	The current user has won and lost games across all game types.	strcmp(currentUser.name, “USER”) == 0	currentUser->games_won_classic == 1 currentUser->games_lost_classic == 1 currentUser->games_won_custom == 2 currentUser->games_lost_custom == 1 currentUser->recentgame[0].path currentUser->recentgame[1].path currentUser->recentgame[2].path	currentUser->games_won_classic == 1 currentUser->games_lost_classic == 1 currentUser->games_won_custom == 2 currentUser->games_lost_custom == 1 currentUser->recentgame[0].path currentUser->recentgame[1].path currentUser->recentgame[2].path	P
viewStatistics	1	The current user has not played any games.	strcmp(currentUser.name, “USER”) == 0	It will display zeroes for all game types and outcomes, and will print nothing for the recent game snapshots.	It will display zeroes for all game types and outcomes, and will print nothing for the recent game snapshots.	P
	2	The current user has played 2 games.	strcmp(currentUser.name, “USER”) == 0	It will display the corresponding values for game types and outcomes, and will print the snapshot of	It will display the corresponding values for game types and outcomes, and will print the snapshot of	P

				both the recent games played.	both the recent games played.	
	3	The current user has played 3 or more games.	strcmp(currentUser.name, “USER”) == 0	It will display the corresponding values for game types and outcomes, and will print the 3 most recent games’ snapshot.	It will display the corresponding values for game types and outcomes, and will print the 3 most recent games’ snapshot.	P
checkCapital	1	name[] contains only capital letters.	strcmp(name, “CAPITAL”) == 0	Return: 1	Return: 1	P
	2	name[] contains only small letters.	strcmp(name, “capital”) == 0	Return: 0	Return: 0	P
	3	name[] contains a combination of capital and small letters.	strcmp(name, “caPiTal”) == 0	Return: 0	Return: 0	P
sortProfiles	1	Users is not alphabetically arranged.	strcmp(users[0].name, “zxy”) == 0 strcmp(users[1].name, “abc”) == 0	strcmp(users[0].name, “abc”) == 0 strcmp(users[1].name, “zxy”) == 0	strcmp(users[0].name, “abc”) == 0 strcmp(users[1].name, “zxy”) == 0	P
	2	Users is alphabetically arranged	strcmp(users[0].name, “abc”) == 0 strcmp(users[1].name, “zxy”) == 0	Nothing happens, since the list is already alphabetically arranged.	Nothing happens, since the list is already alphabetically arranged.	P
	3	Users does not contain any profiles.	num == 0	Nothing happens, since there are no profiles to sort.	Nothing happens, since there are no profiles to sort.	P

checkProfiles	1	There are no existing profiles.	First line in the USER_DIR is 0	It will not print any profiles. Return: 0	It will not print any profiles. Return: 0	P
	2	There are existing profiles.	First line in the USER_DIR is 2 and there are 2 profiles.	It will print the names of the 2 profiles. Return: 2	It will print the names of the 2 profiles. Return: 2	P
	3	There is exactly one profile.	First line in the USER_DIR is 1 and there is 1 profile.	It will print the name of the profile. Return: 1	It will print the name of the profile. Return: 1	P
selectProfile	1	The inputted profile does not exist.	strcmp(path, “user.txt”) == 0	It will print a message telling the player that the profile does not exist. Return: 0	It will print a message telling the player that the profile does not exist. Return: 0	P
	2	The input of the user is not all uppercase letters.	strcmp(name, “user”) == 0	The checkCapital(name) will return 0 and it will print a message telling the plate that the name is not all uppercase letters. Return: 0	The checkCapital(name) will return 0 and it will print a message telling the plate that the name is not all uppercase letters. Return: 0	P
	3	The inputted profile of the user exists.	strcmp(name, “PLAYER”) == 0	The currentUser will become the selected profile and the statistics of that user	The currentUser will become the selected profile and the statistics of that user	P

				will be fetched. Return: 1	will be fetched. Return: 1	
newProfile	1	There are already 10 profiles.	num == 10	It will print a message telling that the max number of profiles have been reached and returns.	It will print a message telling that the max number of profiles have been reached and returns.	P
	2	The inputted name of the user is not all uppercase letters.	strcmp(name, “user”) == 0	The checkCapital(name) will return 0 and it will print a message telling the plate that the name is not all uppercase letters.	The checkCapital(name) will return 0 and it will print a message telling the plate that the name is not all uppercase letters.	
	3	The number of profiles is still below the maximum, but the profile already exists.	strcmp(name, “USER”) == 0	It will print a message telling the user that the profile already exists.	It will print a message telling the user that the profile already exists.	P
	4	The number of profiles is still below the maximum, the inputted name is valid, and the profile does not exist.	strcmp(name, “PLAYER”) == 0	It will create a profile with that name. A text file containing the profile’s statistics will be created, the user directory will include that profile, and the recent game snapshots of the profile will be created and will contain 0.	It will create a profile with that name. A text file containing the profile’s statistics will be created, the user directory will include that profile, and the recent game snapshots of the profile will be created and will contain 0.	P

deleteProfile	1	The inputted name of the user is not all uppercase letters.	strcmp(name, “user”) == 0	It will print a message telling the user that the name is not all uppercase.	It will print a message telling the user that the name is not all uppercase.	P
	2	The inputted profile does not exist.	strcmp(name, “USER”) == 0	It will print a message telling the user that the profile does not exist.	It will print a message telling the user that the profile does not exist.	P
	3	The inputted profile exists and is the current user.	strcmp(name, “PLAYER”) == 0	It will delete the profile. The text file containing the profile’s statistics will be deleted, the user directory will be rewritten without that profile, and the recent game snapshots of the profile will be deleted. The current user will be set to blank.	It will delete the profile. The text file containing the profile’s statistics will be deleted, the user directory will be rewritten without that profile, and the recent game snapshots of the profile will be deleted. The current user will be set to blank.	P
changeProfile	1	The selection of the user is select profile.	choice == 0	The screen will be cleared and the selectProfile function will be called.	The screen will be cleared and the selectProfile function will be called.	P
	2	The selection of the user is new profile.	choice == 1	The screen will be cleared and the newProfile function will be called.	The screen will be cleared and the newProfile function will be called.	P
	3	The selection of the user is delete profile.	choice == 2	The screen will be cleared and the deleteProfile function	The screen will be cleared and the deleteProfile function	P

				will be called.	will be called.	
sortLeaderboard	1	New game WON snapshot is made	temp_ranking[0].time == 12 ranking[0].time == 4 temp_ranking[0] > ranking[0] time	The function will sort the data in ranking[] accordingly.	The function will sort the data in ranking[] accordingly.	P
	2	Existing players do not have any classic games won	none	The function will sort nothing	The function will sort nothing	P
makeLeaderboard	1	Existing players have won classic easy games	strcmp(users[i].recentgame[j].outcome, "WON") == 0 strcmp(users[i].recentgame[j].mode, CLASSIC_EASY) == 0	The function copies won game data into easy leaderboard and compares to other won games, then proceeds to sort	The function copies won game data into easy leaderboard and compares to other won games, then proceeds to sort	P
	2	Existing plates do not have any classic games won	strcmp(users[i].recentgame[j].outcome, "WON") != 0	The function does not write into any leaderboard	The function does not write into any leaderboard	P
	3	Existing players have won classic difficult games	strcmp(users[i].recentgame[j].outcome, "WON") == 0 strcmp(users[i].recentgame[j].mode, CLASSIC_DIFFICULT == 0	The function copies won game data into difficult leaderboard and compares to other won games, then proceeds to sort	The function copies won game data into difficult leaderboard and compares to other won games, then proceeds to sort	P
playCustom	1	Player attempts to play an	filename[] ==	The function will load	The function will load	P

		existing level	game.txt	level then allow user to play with specified level	level then allow user to play with specified level	
	2	Player attempts to play a non-existing level	filename[] == dosnt.txt	The function will print “Level does not exist. Try again.” then prompts user to menu	The function will print “Level does not exist. Try again.” then prompts user to menu	P
playClassic	1	Player decides to play easy	classicSelect == 0	level->cols = 8 level->rows = 8 level->mines = 10 The function makes the board then makes player play the game proper	level->cols = 8 level->rows = 8 level->mines = 10 The function makes the board then makes player play the game proper	P
	2	Player decides to play difficult	classicSelect == 1	level->cols = 15 level->rows = 10 level->mines = 35 The function makes the board then makes player play the game proper	level->cols = 15 level->rows = 10 level->mines = 35 The function makes the board then makes player play the game proper	P
	3	Player decides to go back	classicSelect == 2	The function will print “You have opted to go back.” then returns to menu	The function will print “You have opted to go back.” then returns to menu	P
play	1	Player decides to play classic gamemode	gameSelect == 0	The function calls playClassic() then updates exit to 1 after	The function calls playClassic() then updates exit to 1 after	P

	2	Player decides to play custom gamemode	gameSelect == 1	The function calls playCustom() then updates exit to 1 after	The function calls playCustom() then updates exit to 1 after	P
	3	Player decides to go back	gameSelect == 2	The function will print “You have opted to go back.” then returns to menu	The function will print “You have opted to go back.” then returns to menu	P
startMenu	1	Player decides to create new profile	selection == 0	The function calls newProfile() then updates valid accordingly	The function calls newProfile() then updates valid accordingly	P
	2	Player decides to select existing profile	selection == 1	The function calls selectProfile() then updates valid accordingly	The function calls selectProfile() then updates valid accordingly	P