Written by: Aaron Nicholas Gumapac

Section: S15B

Property...Property is a Monopoly-like game that is exclusively written in C

Players take turns rolling dice and in doing so, move about a board. Each board position is home to a unique property with its very own interactions and UI elements.

Players win once certain conditions are met. By default the only condition required is the bankruptcy of the player's opponent but further customizability is provided.

Installation

Clone this repo

```
git clone https://github.com/aarnich/Property...Property.git
```

Run the makefile

```
cd Property...Property/ && make
```

To reset the build, clean the directory

```
make clean
```

Run the bash script which sets correct environment variables

```
bash runGame.sh
```

Function Unit Testing

Function Name: getRandNum()

Description: Returns a random number within a given range.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	default case for dice rolls, static seed	min = 1, max = 6	5	5	P
2	repeated test, same static seed	min = 1, max = 6	5	5	P
3	tested over 10^4 iterations, new range	min = 1, max = 9	no numerical pattern emerged	no numerical pattern emerged	P

Function Name: getAllPlayerProperties()

Description: Returns a string that contains every property the player owns.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	every property you can own is owned by the bank: default case	statekey = 444949444, playerID = 1	" "	" "	P
2	player 1 owns the first 3 properties	statekey = 444949555, playerID = 1	"Tree house Electric co. Beach house"	"Tree house Electric co. Beach house"	P
3	player 2 owns the last 3 properties	statekey = 666949444, playerID = 2	"Railroad Igloo Farm house"	"Railroad Igloo Farm house"	P

Function Name: getDigits()

Description: Returns the number of digits in a number

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	count how many digits in a 3 digit integer	333	3	3	P
2	0 digits	0	0	0	P
3	statekey	444949444	9	9	P

Function Name: playerOwnsProperties()

Description: Returns true if player owns any properties, false otherwise.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case, player 1 does not own any properties	statekey = 444949444, playerID = 1	false	false	P
2	player 2 owns every property, checking for player 1	statekey = 666969666, playerID = 1	false	false	P
3	player 1 owns 3 properties, player 2 owns 3 properties, checking for player 2	statekey = 666949555, playerID = 2	true	true	P

Function Name: playerDialogue()

Description: Prompts the user with a message. Returns true if player selects the first choice in the selection, false otherwise (signalling end turn).

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	prompt for property purchase, player ends turn	dialogueMessage = "Buy\nEnd Turn", validInputs = "BE", choice = 'E'	false	false	P
2	prompt for renovation, player renovates	dialogueMessage = "Renovate\nEnd Turn", validInputs = "RE", choice = 'R'	true	true	P

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
3	prompt for property purchase, player buys	dialogueMessage = "Buy\nEnd Turn", validInputs = "BE", choice = 'B'	true	true	P

Function Name: playerOwns()

Description: Returns true if player owns the current property, false otherwise.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	player 2 owns current property, checking for player 1	propIndicator = 6, playerID = 1	false	false	P
2	player 1 owns current property	propIndicator = 5, playerID = 1	true	true	P
3	player 1 owns a renovated property	propIndicator = 7, playerID = 1	true	true	P

Function Name: getPlayerSellChoice()

Description: Returns the statekey index of the property the player would like to sell only if the player owns said property.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	player 1 sells the fifth property on the statekey	statekey = 666979666, playerID = 1, choice = 5	5	5	P
2	player 1 sells the fifth property on the statekey	statekey = 666969666, playerID = 1, choice = 5	"You do not own that property"	"You do not own that property"	P
3	player 2 sells the ninth property on the statekey	statekey = 666969666, playerID = 2, choice = 9	9	9	P

Function Name: fetchPlayerName()

Description: Changes player.name variable after prompting the user for an input.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	player sets name as "Trump"	input = "Trump"	player.name = "Trump"	player.name = "Trump"	P
2	scanf char* only captures before the first " ", enter with last name	input = "Trumpy Donaldy"	player.name = "Trumpy"	player.name = "Trumpy"	P
3	player sets name only as " "	input = " "	"invalid name, try again"	"invalid name, try again"	P

Function Name: isPrime()

Description: Returns true if given integer is prime, false otherwise. Integer is prime if and only if it is divided by exactly 2 integers with no remainders.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	check if 1 is a prime according to definition	num = 1	false	false	P
2	check if 2 is a prime	num = 2	true	true	P
3	check if 0 is a prime	num = 0	false	false	P

Function Name: normalizeNumByIndex()

Description: Returns an integer whose digits within the index position has been set to zero.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	get ten thousands	num = 12345, index = 3	12000	12000	P
2	get tens	num = 12, index = 1	10	10	P
3	get hundreds	num = 321, index = 2	300	300	P

Function Name: handleInput()

Description: Prompts the user to enter a char and capitalizes char if necessary. Returns said char if and only if it is within the set of valid characters.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	check if 'o' is within validinputs	input = 'o', validInputs = "NO"	'O'	'O'	P
2	user enters '1'	input = '1', validInputs = "2345"	"invalid input, try again"	"invalid input, try again"	P
3	user enters more than one digit	input = "what" validInputs = "abcd"	"invalid input, try again"	"invalid input, try again"	P

Function Name: readStatekeyAtIndex()

Description: Returns integer value + offset at the given statekey index.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	get the first number	statekey = 444949444, index = 1, offset = -4	0	0	P
2	get a number outside of the statekey	statekey = 444949444, index = 10, offset= - 4	"invalid index" returns 0	"invalid index" returns 0	P
3	get the ninth number	statekey = 144949444, index = 9, offset = -4	-1	-1	P

Function Name: mutateStatekeyAtIndex()

Description: Replace a statekey digit (value - offset) given the index and returns the mutated statekey.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	set the first digit to 2	statekey = 444949444, index = 1, value = 2	444949446	444949446	P

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
2	mutate a digit not inside the statekey	statekey = 444949444, index = 0, value = 2	"invalid index" returns 0	"invalid index" returns 0	P
3	sets the ninth digit to 4	statekey = 444949444, index = 9, value = 4	844949444	844949444	P

Function Name: getPropertyCost()

Description: Calculates the purchase cost of a property given its position on the board.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	cost of the tree house	propIndex = 1	20	20	P
2	cost the railroad	propIndex = 7	railCost (default = 100)	railCost (default = 100)	P
3	cost of the electric company	propIndex = 2	electricCost (default = 150)	electricCost (default = 150)	P

Function Name: getRent()

Description: Returns the cost of rent for the property at index.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	rent for tree house	propIndex = 1	4	4	P
2	rent for railroad	propIndex = 7	35	35	P
3	player rolls a 2	propIndex = 2	electricMulti * roll (default = 16)	16	P

Function Name: exponentiateNum()

Description: raises a base to a power, does not cover fractional outputs.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	raise 10 to 2	base = 10, power = 2	100	100	P
2	raise 10 to 0	base = 10, power = 0	1	1	P
3	raise 10 to 1	base = 10, power = 1	10	10	P

Function Name: getValidInteger()

Description: Error proof funcion that prompts the user for an integer value and only returns if input is valid. Does not accept non numeric types.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	user inputs 2	input = 2	2	2	P
2	user inputs non numeric type	'C'	"invalid input, try again"	"invalind input, try again"	P
3	user inputs floating point number	0.9	0	0	P

Function Name: checkIfInRange()

Description: Returns true if a given integer is within a given range, false otherwise.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	check if 1 is inside 1,10	min = 1, max = 10, input = 1	true	true	P
2	check if 0 is inside 5,10	min = 5, max = 10, input = 0	false	false	P
3	check for 0 inside 0,0	min = 0, max = 0, input = 0	true	true	P

Function Name: editRange()

Description: Returns a new range with user-modified min and max vals.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	range.min = 5, range.max = 10	input.min = 6	range.min = 6, range.max = 10	range.min = 6, range.max = 10	P
2	min is greater than max	input.min = 10, input.max = 2	"invalid range, try again"	"invalid range, try again"	P
3	min = max	input.min = 5, input.max = 5	range.min = 5, range.max = 5	range.min = 5, range.max = 5	P

Function Name: continueGame()

Description: Returns true if winner == NONE, false otherwise.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	players tie	winner = TIE	false	false	P
2	player 1 wins	winner = PLAYER1	false	false	P
3	base case	winner = NONE	true	true	P

Function Name: initializePlayer()

Description: creates a Player struct with initial values, defaults are defined in header

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case		initPlayer	initPlayer	P
2	change default pos value to 2	read DEFAULT_STARTINGPOS	initPlayer.pos = 2	initPlayer.pos = 2	P
3	change default balance to 200	read DEFAULT_STARTINGBALANCE	initPlayer.balane = 200	initPlayer.balance = 200	P

Function Name: initializeSettings()

Description: Returns a settings struct with initial values, defaults are defined in header.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case		initSettings	initSettings	P

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
2	change railroad rent to 10	read DEFAULT_RAILRENT	initSettings.railRent = 10	initSettings.railRent = 10	P
3	change electric co. multiplier to 10	read DEFAULT_ELECTRIC_MULTI	initSettings.electricMulti = 10	initsettings.eletricMulti = 10	P

Function Name: initializeWinstate()

Description: Returns a winstate struct with initial values. Defaults are hardcoded.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case		initWinstate	initWinstate	P
2	set winner to player 1		initWinstate.winner = PLAYER1	initWinstate.winner = PLAYER1	P
3	set winRationale[0] to ENEMY_BANKRUPTCY		initWinstate.winRationale[0] = ENEMY_BANKRUPTCY	initWinstate.winRationale[0] = ENEMY_BANKRUPTCY	P

Function Name: initializeWinconditions()

Description: Returns a winconditions struct with initial values. Defaults are hardcoded.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case		initWinconditions	initWinconditions	P
2	use in initializeSettings()	initSettings.winconds = initializeWinconditions()	initWinconditions	initWinconditions	P
3	use in displayWinconds()	displayWinconds(initializeWinconditions())	display default winconditions	display default winconditions	P

Function Name: initializeGame()

Description: Returns a gamepkg struct with initial values, serves as the primary initializer for the game.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case		game	game	P
2	remove initSettings()		game.SETTINGS = garbage values	game.SETTINGS = garbage values	P
3	change statekey to 000000000	read STARTING_STATEKEY	game.STATEKEY = 0	game.STATEKEY = 0	P

Function Name: populateContext()

Description: Populates the wincontext list in order to fill indexes equal to NOCONTEXT.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	insert ENEMY_BANKRUPTCY to empty list	context = ENEMY_BANKRUPTCY	arrContext[0] = ENEMY_BANKRUPTCY	arrContext[0] = ENEMY_BANKRUPTCY	P

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
2	insert WINNING_BALANCE_REACHED to list with index 0 filled	context = WINNING_BALANCE_REACHED	arrContext[1] = ENEMY_BANKRUPTCY	arrContext[1] = ENEMY_BANKRUPTCY	P
3	attempt to insert LOSING_BALANCE_REACHED to list with all indexes filled	context = LOSING_BALANE REACHED	(no change) arrContext	(no change) arrContext	P

Function Name: updatePlayer()

Description: Updates activePlayer and player jail statuses given their memory locations.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case, player 1's turn is done	activePlayer = 1	activePlayer = 2	activePlayer = 2	P
2	player 2 is jailed, player 1's turn is next	p2_isJailed = true, activePlayer = 2	p2_isJailed = false, activePlayer = 1	p2_isJailed = false, activePlayer = 1	P
3	both players are jailed, player 1's turn just ended	p2_jsJailed = true, p1_isJailed = true, activePlayer = 1	p2_jsJailed = false, p1_isJailed = false, activePlayer = 4	p2_jsJailed = false, p1_isJailed = false, activePlayer = 4	P

Function Name: updateWinstate()

Description: Returns a new winstate struct given win settings and new player values.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	player 2 satifies both winconditions, player 1 only satisfies one	p1Flags = 1, p2Flags = 2	winstate.winner = PLAYER2	winstate.winner = PLAYER2	P
2	both players satisfy the win condition	p1Flags = 1, p2Flags = 1	winstate.winner = TIE	winsstate.winner = TIE	P
3	base case	p1Flags = 0. p1Flags = 0	winstate.winner = NONE	winsate.winner = NONE	P

Function Name: saveGame()

Description: Returns a new gamepkg struct that stores new player values and updated winstate values.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	base case	player1, player2, playerkeys, gamestate	updatedGame	updatedGame	P
2	player 1's key is 1	player1, player2, playerkeys, gamestate	updatedGame.arrPlayers[1] = player1	updatedGame.arrPlayers[1] = player1	P
3	player 1's key is 0	player1, player2, playerkeys, gamestate	updatedGame.arrPlayers[0] = player1	updatedGame.arrPlayers[0] = player1	P

Function Name: updatedPlayerPosition()

Description: Returns a player struct with new position values and balance changes if player crosses the 0th position.

#	Test Description	Sample Input/Arguments	Expected Output	Actual Output	P/F
1	PLayer 1 at pos 1 rolls a 5	player.pos = 1, roll = 5	player.pos = 6	player.pos = 6	P
2	Player 1 at pos 5 rolls a 9	player.pos = 5, roll = 9	player.pos = 4, player.balance += goBonus	player.pos = 4, player.balance += goBonus	P
3	PLayer 2 at pos 0 rolls a 4	player.pos = 0, roll = 4	player.pos = 4	player.pos = 4	P

References

- Computerphile. YouTube. Retrieved February 4, 2022, from https://www.youtube.com/user/Computerphile/
- SICP. Structure and interpretation of computer programs. Retrieved February 4, 2022, from https://mitpress.mit.edu/sites/default/files/sicp/full-text/book/book.html