LOGIC FORMULATION AND INTRODUCTORY PROGRAMMING [CCPROG1]

Patrick James T. Marcellana
S20A

Machine Project

Test Script

Function Name	#	Test Description	Sample Input	Expected Result	Actual Result	P/F
isChoiceValid ()	1	The formal parameter, pChoice, is a symbol character.	pChoice = '&'	Since this input is invalid, the function should keep asking the player to enter a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function kept asking the player to enter a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
	2	The formal parameter, pChoice, is a single-digit number character.	pChoice = '0'	Since this input is invalid, the function should keep asking the player to enter a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function kept asking the player to enter a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
	3	The formal parameter, pChoice, is a valid lowercase letter character (a or b).	pChoice = 'a'	The loop should end, and the actual parameter should update its value to 'a'.	The loop ended, and the actual parameter updated its value to 'a'.	P
getStartChoice ()	1	The formal parameter, pStartChoice, is a valid uppercase letter character (A or B).	pStartChoice = 'A'	The actual parameter, cStartChoice, should update its value to 'A'.	The actual parameter, cStartChoice, updated its value to 'A'.	P
	2	The formal parameter, pStartChoice, is a valid lowercase letter character (a or b).	pStartChoice = 'b'	The actual parameter, cStartChoice, should update its value to 'b'.	The actual parameter, cStartChoice, updated its value to 'b'.	P
	3	The formal parameter, pStartChoice, is a single-digit number character.	pStartChoice = '5'	Since this is an invalid input, the function, isChoiceValid (), should place the game into a loop until the player enters a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function, isChoiceValid (), kept asking the player to enter [A] or [B] until the player entered a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
displayPrompt ()	1	The input for nPromptContinue is a valid single-digit integer (1).	nPromptContinue = 1	No expected text output. The game should continue on its progress.	The game continued on its progress.	P
	2	The input for nPromptContinue is an invalid single-digit integer.	nPromptContinue = 5	The function should keep asking the player to enter a valid input. Expected Text Output: "Invalid input. Press [1] then Enter to Continue: "	The function kept asking the player to enter a valid input. Actual Text Output: "Invalid input. Press [1] then Enter to Continue: "	P
	3	The input for nPromptContinue is an invalid five-digit integer.	nPromptContinue = 21412	The function should keep asking the player to enter a valid input. Expected Text Output: "Invalid input. Press [1] then Enter to Continue: "	The function kept asking the player to enter a valid input. Actual Text Output: "Invalid input. Press [1] then Enter to Continue: "	P
getRandomPrice ()	1	The formal parameters, nMinPrice and nMaxPrice, are passed with values based on Daisy Mae's selling price range.	nMinPrice = 90 nMaxPrice = 110	The function should return a random number ranging from 90 to 110, inclusive.	The function returned the randomized number, 105.	P
	2	The formal parameters, nMinPrice and nMaxPrice, are passed with values that are four-digit integers.	nMinPrice = 1000 nMaxPrice = 2000	The function should returrn a random number ranging from 1000 to 2000, inclusive.	The function returned the randomized number, 1041.	P
	3	The formal parameters, nMinPrice and nMaxPrice, are passed with values that are six-digit integers.	nMinPrice = 200000 nMaxPrice = 500000	The function should return a random number ranging from 200000 to 500000, inclusive.	The function returned the randomized number, 224861.	P
	1	The formal parameter, pBuyChoice, is a valid uppercase letter character (A or B).	pBuyChoice = 'B'	The actual parameter, cBuyChoice, should update its value to 'B'.	The actual parameter, cBuyChoice, updated its value to 'B'.	P

getBuyChoice ()	2	The formal parameter, pBuyChoice, is a valid lowercase letter character (a or b).	pBuyChoice = 'b'	The actual parameter, cBuyChoice, should update its value to 'b'.	The actual parameter, cStartChoice, updated its value to 'b'.	P
	3	The formal parameter, pBuyChoice, is a symbol character.	pBuyChoice = '%'	Since this is an invalid input, the function, isChoiceValid (), should place the game into a loop until the player enters a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function, isChoiceValid (), kept asking the player to enter [A] or [B] until the player entered a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
computeBuyCost ()	1	The formal parameter, nCurrentStacks is passed with a single-digit integer, and the formal parameter, nBuyStackPrice is passed with a value less than 1000.	nCurrentStacks = 3 nBuyStackPrice = 930	The function should return a value of 2790.	The function returned a value of 2790.	P
	2	The formal parameter, nCurrentStacks is passed with a double-digit integer, and the formal parameter, nBuyStackPrice is passed with a value equal to 1000.	nCurrentStacks = 11 nBuyStackPrice = 1040	The function should return a value of 11440.	The function returned a value of 11440.	P
	3	The formal parameter, nCurrentStacks is passed with a value of 0, and the formal parameter, nBuyStackPrice is passed with a value equal to 1000.	nCurrentStacks = 0 nBuyStackPrice = 1000	The function should return a value of 0.	The function returned a value of 0.	P
isStacksBoughtValid ()	1	The formal parameter, pCurrentStacks, is a negative number. The formal parameter, nPlayerBells is passed with the value equal to the starting number of bells of the player, while nBuyStackPrice is passed with a value less than 1000.	pCurrentStacks = -5 nPlayerBells = 5000 nBuyStackPrice = 900	Since the value of pCurrentStacks is negative, the function should keep asking the player to enter an input that is greater than or equal to 0. Expected Text Output: "Invalid input. You can't buy negative stacks of turnips! Input the number of stacks that you want to buy: "	The function kept asking the player to enter an input that is greater than or equal to 0. Actual Text Output: "Invalid input. You can't buy negative stacks of turnips! Input the number of stacks that you want to buy: "	P
	2	The formal parameter, pCurrentStacks, is a two-digit integer. The formal parameter, nPlayerBells is passed with a five-digit integer value, while nBuyStackPrice is passed with a value greater than 1000.	pCurrentStacks = 25 nPlayerBells = 10500 nBuyStackPrice = 1040	Since the cost to buy the number of stacks (pCurrentStacks * nBuyStackPrice) is more than the player's current bells (nPlayerBells), the function should keep asking the player to enter an input that is enough with his or her current number of bells. Expected Text Output: "You can't afford that many! You only have 10500 bells. Input the number of stacks that you want to buy: "	The function kept asking the player to enter an input that is enough with his or her current number of bells. Actual Text Output: "You can't afford that many! You only have 10500 bells. Input the number of stacks that you want to buy: "	P
	3	The formal parameter, pCurrentStacks, is equal to 0. The formal parameter, nPlayerBells is passed with a six-digit integer value, while nBuyStackPrice is passed with a value equal to 1000.	pCurrentStacks = 0 nPlayerBells = 200000 nBuyStackPrice = 1000	The loop should end, and the actual parameter, nCurrentStacks should retain / update its value to 0.	The loop ended, and the actual parameter, nCurrentStacks retained / updated its value to 0.	P
	1	The formal parameter, pCurrentStacks, is greater than 0. The formal parameter, nPlayerBells is passed with a value equal to the starting number of bells of the player, while nBuyStackPrice is passed with a value less than 1000.	pCurrentStacks = 5 nPlayerBells = 5000 nBuyStackPrice = 940	The actual parameter, nCurrentStacks, should update its value to 5.	The actual parameter, nCurrentStacks, updated its value to 5.	P

getBoughtStacks ()	2	The formal parameter, pCurrentStacks, is less than 0. The formal parameter, nPlayerBells is passed with a five-digit integer value, while nBuyStackPrice is passed with a value equal to 1000.	pCurrentStacks = -20 nPlayerBells = 23450 nBuyStackPrice = 1000	Since the input is a negative integer, the function, isStacksValid (), should place the game into a loop until the player enters an input that is greater than or equal to 0. Expected Text Output: "Invalid input. You can't buy negative stacks of turnips! Input the number of stacks that you want to buy: "	The function, isStacksValid (), told the player that he or she can't buy a negative number of stacks. It kept asking the player to enter an input that is greater than 0. Actual Text Output: "Invalid input. You can't buy negative stacks of turnips! Input the number of stacks that you want to buy: "	P
	3	The formal parameter, pCurrentStacks, is a two-digit integer. The formal parameter, nPlayerBells is passed with a four-digit integer value, while nBuyStackPrice is passed with a value greater than 1000.	pCurrentStacks = 10 nPlayerBells = 6300 nBuyStackPrice = 1100	Since the buy cost would be greater than the player's bells, the function, isStacksValid (), should place the game into a loop until the player enters an input that he or she can afford with his or her current bells. Expected Text Output: "You can't afford that many! You only have 6300 bells. Input the number of stacks that you want to buy: "	The function, isStacksValid (), told the player that he or she can't afford that many stacks. It kept asking the player to enter an input that he or she can afford with his or her current bells. Actual Text Output: "You can't afford that many! You only have 6300 bells. Input the number of stacks that you want to buy: "	P
getWeeklyTrend ()	1	The formal parameter, nBuyPrice, is passed with a value less than 100. The local variable, nWeeklyTrend, gets a randomized value of 0.	nBuyPrice = 92 nWeeklyTrend = 0	Since nWeeklyTrend having a value of 0 means a bad trend, the actual parameters, nMinSellPrice and nMaxSellPrice, should update their values to 20 and 92, respectively.	The actual parameters, nMinSellPrice and nMaxSellPrice, updated their values to 20 and 92, respectively.	P
	2	The formal parameter, nBuyPrice, is passed with a value equal to 100. The local variable, nWeeklyTrend, gets a randomized value of 1.	nBuyPrice = 100 nWeeklyTrend = 1	Since nWeeklyTrend having a value of 1 means an average trend, the actual parameters, nMinSellPrice and nMaxSellPrice, should update their values to 80 and 105, respectively.	The actual parameters, nMinSellPrice and nMaxSellPrice, updated their values to 80 and 105, respectively.	P
	3	The formal parameter, nBuyPrice, is passed with a value more than 100. The local variable, nWeeklyTrend, gets a randomized value of 2.	nBuyPrice = 105 nWeeklyTrend = 2	Since nWeeklyTrend having a value of 2 means an awesome trend, the actual parameters, nMinSellPrice and nMaxSellPrice, should update their values to 105 and 315, respectively.	The actual parameters, nMinSellPrice and nMaxSellPrice, updated their values to 105 and 315, respectively.	P
getSellChoice ()	1	The formal parameter, pSellChoice, is a single-digit number character.	pSellChoice = '9'	Since this is an invalid input, the function, isChoiceValid (), should place the game into a loop until the player enters a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function, isChoiceValid (), kept asking the player to enter [A] or [B] until the player entered a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
	2	The formal parameter, pSellChoice, is a symbol character.	pSellChoice = '@'	Since this is an invalid input, the function, isChoiceValid (), should place the game into a loop until the player enters a valid input. Expected Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	The function, isChoiceValid (), kept asking the player to enter [A] or [B] until the player entered a valid input. Actual Text Output: "That's not a valid input, mate! Press [A] or [B], then Enter: "	P
	3	The formal parameter, pSellChoice, is a valid lowercase letter character (a or b).	pSellChoice 'a'	The actual parameter, cSellChoice, should update its value to 'a'.	The actual parameter, cSellChoice, updated its value to 'a'.	P
	1	The formal parameter, pStacksSold, is a negative integer, and the formal parameter, nCurrentStacks, is passed with a double-digit integer value.	pStacksSold = -3 nCurrentStacks = 10	Since the value is negative, the function should keep asking the player to enter an input that is greater than or equal to 0. Expected Text Output: "You can't sell negative number of stacks! Input the number of stacks you want to sell: "	The function kept asking the player to enter an input that is greater than or equal to 0. Actual Text Output: "You can't sell negative number of stacks! Input the number of stacks you want to sell: "	P

isStackSoldValid ()	2	The formal parameter, pStacksSold, is a double-digit integer, and the formal parameter, nCurrentStacks, is passed with a single-digit integer value.	pStacksSold = 14 nCurrentStacks = 4	Since the number of stacks entered to be sold is more than the current stacks that the player has, the function should keep asking the player to enter an input (pStacksSold) that is less than or equal to nCurrentStacks. Expected Text Output: "You don't have that many stacks! I nput the number of stacks you want to sell: "	The function kept asking the player to enter an input (pStacksSold) that is less than or equal to nCurrentStacks. Actual Text Output: "You don't have that many stacks! Input the number of stacks you want to sell: "	P
	3	The formal parameter, pStacksSold, is less than the formal parameter, nCurrentStacks.	pStacksSold = 5 nCurrentStacks = 8	The loop should end, and the actual parameter, pStacksSold, should update its value to 5.	The loop ended, and the actual parameter, pStacksSold, updated its value to 5.	P
getSoldStacks ()	1	The formal parameter, pStacksSold, is greater than 0. The formal parameter, nCurrentStacks, is passed with a value greater than 10.	pStacksSold = 3 nCurrentStacks = 16	The actual parameter, nStacksSold, should update its value to 3.	The actual parameter, nStacksSold, updated its value to 3.	P
	2	The formal parameter, pStacksSold, is less than 0. The formal parameter, nCurrentStacks is a single-digit integer.	pStacksSold = -5 nCurrentStacks = 2	Since the input is a negative integer, the function, isStackSoldValid (), should place the game into a loop until the player enters an input that is greater than or equal to 0 Expected Text Output: "You can't sell negative number of stacks! Input the number of stacks you want to sell: "	"The function, isStackSoldValid (), told the player that he or she can't sell a negative number of stacks. It kept asking the player to enter an input that is greater than or equal to 0. Actual Text Output: "You can't sell negative number of stacks! Input the number of stacks you want to sell: "	P
	3	The formal parameters, pStacksSold and nCurrentStacks, are greater than 100.	pStacksSold = 110 nCurrentStacks = 140	The actual parameter, nStacksSold, should update its value to 110.	The actual parameter, nStacksSold, should update its value to 110.	P