CCPROG1 Term 1, AY 2021 – 2022

Test Script Document

Name	Eugene Guillermo
Section	S17

Test Scripts

There should be at least 3 distinct test classes (as indicated in the description) per function. There is no need to test functions which are only for screen design.

Function Name: `getRandomNumber`					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	nLow and nUp are positive	nLow = 1 nUp = 2	1 to 2	2	P
2	nLow is negative and nUp is positive	nLow =-1 nUp = 1	-1 to 1	1	P
3	nLow is higher than nUp	nLow = 2 nUp = 1	Error	Nothing	F

Function Name: addInterest					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	fInterestRate and fGD are positive	fInterestRate = 1 fGD = 1	2.0	2.0	P
2	fInterestRate is positive and fGD is negative	fInterestRate = 1 fGD = -1	-2.0	-2.0	P
3	fInterestRate is negative and fGD is positive	fInterestRate = -1 fGD = 1	0.0	0.0	P

Function Name: `listPrices`					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	nCurrLoc is a valid region and nPrices is an array to be stored in	nCurrLoc = 1 nPrices = nPrices[8]	Print Winterfell nPrices[0] = 180 to 200 nPrices[1] = 200 to 250	Print Winterfell nPrices[0] = 196 nPrices[1] = 222 nPrices[2] = 396	P

			nPrices[2] = 380 to 400 nPrices[3] = 480 to 500 nPrices[4] = 580 to 600 nPrices[5] = 680 to 700 nPrices[6] = 780 to 800 nPrices[7] = 880 to 900	nPrices[3] = 496 nPrices[4] = 596 nPrices[5] = 696 nPrices[6] = 796 nPrices[7] = 896	
2	nCurrLoc is not a valid region and nPrices is an array to be stored in	nCurrLoc = 10 nPrices = nPrices[8]	nPrice = {0, 0, 0, 0, 0, 0, 0} (initialized in main function)	nPrices[0] = 0 nPrices[1] = 0 nPrices[2] = 0 nPrices[3] = 0 nPrices[4] = 0 nPrices[5] = 0 nPrices[6] = 0 nPrices[7] = 0	P
3	nCurrLoc is a valid region and nprice is an array to be stored in	nCurrLoc = 1 nPrices = nprice[8] nPrices[8] (initialized in main function)	Print Winterfell nprice[0] = 180 to 200 nprice[1] = 200 to 250	nprice[0] = 199 nPrices[0] = 0 nprice[1] = 216 nPrices[1] = 0 nprice[2] = 399 nPrices[2] = 0	P

			nprice[2] = 380 to 400 nprice[3] = 480 to 500 nprice[4] = 580 to 600 nprice[5] = 680 to 700 nprice[6] = 780 to 800 nprice[7] = 880 to 900 nPrices = {0, 0, 0, 0, 0, 0, 0, 0}	nprice[3] = 499 nPrices[3] = 0 nprice[4] = 599 nPrices[4] = 0 nprice[5] = 699 nPrices[5] = 0 nprice[6] = 799 nPrices[6] = 0 nprice[7] = 899 nPrices[7] = 0	

Function Name: actionBuy					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	All positive values	fGD = 1000.0 nMaxCap = 50 nCurrCap = 0 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = 700.0 nMaxCap = 50 nCurrCap = 3 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 3, 0, 0, 0, 0, 0}	fGD = 700.000000 nMaxCap = 50 nCurrCap = 3 nPrices[0] = 100 nPrices[1] = 100 nPrices[2] = 100 nPrices[3] = 100 nPrices[4] = 100	P

				nPrices[5] = 100 nPrices[6] = 100 nPrices[7] = 100 nItems[0] = 0 nItems[1] = 0 nItems[2] = 3 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	
2	All positive except amount of GDs	fGD = -1000.0 nMaxCap = 50 nCurrCap = 0 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = -1000.0 nMaxCap = 50 nCurrCap = 0 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0}	fGD = -1000.000000 nMaxCap = 50 nCurrCap = 0 nPrices[0] = 100 nPrices[1] = 100 nPrices[2] = 100 nPrices[3] = 100 nPrices[4] = 100 nPrices[5] = 100 nPrices[6] = 100	P

				nPrices[7] = 100 nItems[0] = 0 nItems[1] = 0 nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	
3	Negative Prices	fGD = 1000.0 nMaxCap = 50 nCurrCap = 0 nPrices[8] = {-100, -100, -100, -100, -100, -100, -100, -100} nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = 1300.0 nMaxCap = 50 nCurrCap = 0 nPrices[8] = {-100, -100, -100, -100, -100, -100, -100, -100} nItems[8] = {0, 0, 3, 0, 0, 0, 0, 0}	fGD = 1300.000000 nMaxCap = 50 nCurrCap = 3 nPrices[0] = -100 nPrices[1] = -100 nPrices[2] = -100 nPrices[3] = -100 nPrices[4] = -100 nPrices[5] = -100 nPrices[6] = -100 nPrices[7] = -100 nItems[0] = 0	P

				nItems[1] = 0 nItems[2] = 3 nItems[3] = 3 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	

Function Name: actionSell					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	Selling items but ItemQuant is the same as the items owned	fGD = 1000.0 nCurrCap = 3 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 3, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = 1300.0 nCurrCap = 0 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0}	fGD = 1300.000000 nCurrCap = 0 nPrices[0] = 100 nPrices[1] = 100 nPrices[2] = 100 nPrices[3] = 100 nPrices[4] = 100 nPrices[5] = 100 nPrices[6] = 100 nPrices[7] = 100 nItems[0] = 0 nItems[1] = 0	P

				nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	
2	Selling items but ItemQuant is more than the items owned	fGD = 1000.0 nCurrCap = 2 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 2, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = 1000.0 nCurrCap = 2 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 2, 0, 0, 0, 0, 0}	fGD = 1000.000000 nCurrCap = 2 nPrices[0] = 100 nPrices[1] = 100 nPrices[2] = 100 nPrices[3] = 100 nPrices[4] = 100 nPrices[5] = 100 nPrices[6] = 100 nPrices[7] = 100 nItems[0] = 0 nItems[1] = 0 nItems[2] = 2 nItems[3] = 0 nItems[4] = 0	P

				nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	
3	Selling items but ItemQuant is less than the items owned	fGD = 1000.0 nCurrCap = 4 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 4, 0, 0, 0, 0, 0} (User inputs inside the function) nItem = 3 nItemQuant = 3	fGD = 1300.0 nCurrCap = 1 nPrices[8] = {100, 100, 100, 100, 100, 100, 100, 100} nItems[8] = {0, 0, 1, 0, 0, 0, 0, 0}	fGD = 1300.0000000 nCurrCap = 1 nPrices[0] = 100 nPrices[1] = 100 nPrices[2] = 100 nPrices[3] = 100 nPrices[4] = 100 nPrices[5] = 100 nPrices[6] = 100 nPrices[7] = 100 nItems[0] = 0 nItems[1] = 0 nItems[2] = 1 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	P

Function Name: actionBank					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	User chooses to quit	fGD = 1000 fSavings = 0 fDebt = 0 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0 nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nBChoice = 9	fGD = 1000 fSavings = 0 fDebt = 0 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 9 nCurrCap = 0 nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0}	fGD = 1000.000000 fSavings = 0.000000 fDebt = 0.000000 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 9 nCurrCap = 0 nItems[0] = 0 nItems[1] = 0 nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	P
2	User chooses to goBack	fGD = 1000 fSavings = 0 fDebt = 0 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0	fGD = 1000 fSavings = 0 fDebt = 0 nDayCounter = 0 nMaxCap = 0	fGD = 1000.000000 fSavings = 0.000000 fDebt = 0.000000	P

		nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nBChoice = 5	nCurrLoc = 1 nCurrCap = 0 nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0}	nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0 nItems[0] = 0 nItems[1] = 0 nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	
3	User chooses to deposit	fGD = 1000 fSavings = 0 fDebt = 0 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0 nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nBChoice = 1 fDep = 1000;	fGD = 0 fSavings = 1000 fDebt = 0 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0 nItems[8] = {0, 0, 0, 0, 0, 0, 0, 0}	fGD = 0.000000 fSavings = 1000.000000 fDebt = 0.000000 nDayCounter = 0 nMaxCap = 0 nCurrLoc = 1 nCurrCap = 0 nItems[0] = 0 nItems[1] = 0	P

				nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	

Function Name: actionWheelhouse					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	User chooses to upgrade and go to another trading partner	fSavings = 0 fDebt = 0 fGD = 1000 nDayCounter = 10 nMaxCap = 50 nCurrLoc = 1 nPrices[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) cUpChoice = 'y' nCurrLoc = 4	fSavings = 0 fDebt = 0 fGD = 800 nDayCounter = 9 nMaxCap = 100 nCurrLoc = 4 nPrices[8] = {180 to 200, 280 to 300, 380 to 400, 480 to 500, 500 to 550, 680 to 700, 780 to 800, 880 to 900}	fSavings = 0.000000 fDebt = 0.000000 fGD = 800.000000 nDayCounter = 9 nMaxCap = 100 nCurrLoc = 4 nCurrCap = 0 nItems[0] = 189 nItems[1] = 289 nItems[2] = 389 nItems[3] = 489 nItems[4] = 539	P

				nItems[5] = 689 nItems[6] = 789 nItems[7] = 889	
2	User chooses to go back	fSavings = 0 fDebt = 0 fGD = 0 nDayCounter = 10 nMaxCap = 50 nCurrLoc = 1 nPrices[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) nCurrLoc = 1 cLocChoice = 'y'	fSavings = 0 fDebt = 0 fGD = 0 nDayCounter = 10 nMaxCap = 50 nCurrLoc = 1 nPrices[8] = {0, 0, 0, 0, 0, 0, 0, 0}	fSavings = 0.000000 fDebt = 0.000000 fGD = 0.000000 nDayCounter = 10 nMaxCap = 50 nCurrLoc = 1 nItems[0] = 0 nItems[1] = 0 nItems[2] = 0 nItems[3] = 0 nItems[4] = 0 nItems[5] = 0 nItems[6] = 0 nItems[7] = 0	P
3	User goes to another trading partner with debt and savings	fSavings = 1000 fDebt = 1000 fGD = 0 nDayCounter = 10 nMaxCap = 50 nCurrLoc = 2	fSavings = 1100 fDebt = 1050 fGD = 0 nDayCounter = 10 nMaxCap = 50	fSavings = 1100.000000 fDebt = 1050.000000 fGD = 0.000000	P

		nPrices[8] = {0, 0, 0, 0, 0, 0, 0, 0} (User inputs inside the function) cLocChoice = 'y' nCurrLoc = 1	nCurrLoc = 1 nPrices[0] = 180 to 200 nPrices[1] = 200 to 250 nPrices[2] = 380 to 400 nPrices[3] = 480 to 500 nPrices[4] = 580 to 600 nPrices[5] = 680 to 700 nPrices[6] = 780 to 800 nPrices[7] = 880 to 900	nDayCounter = 9 nMaxCap = 50 nCurrLoc = 1 nItems[0] = 195 nItems[1] = 203 nItems[2] = 395 nItems[3] = 495 nItems[4] = 595 nItems[5] = 695 nItems[6] = 795 nItems[7] = 895	