Description: AllowIt is a budgeting program that takes the allowance, budget plan, and expenses of the user, and evaluates these values to guide the user towards financial stability.
Programmed by: Roimarc Bilbao | S20A
Last modified: December 4, 2022
Version: 2.4.2
Acknowledgements: patorjk Text to ASCII Art Generator (https://patorjk.com/software/taag/)

Function Name	#	Test Description	Sample Input	Expected Result	Actual Result	Pass/Fail
getAllowance	1	Enter a negative value for allowance	*fAllowance = -3	“Invalid Input.”, *fAllowance = 0, then prompts for another input	“Invalid Input.”, *fAllowance = 0, then prompts for another input	Pass
	2	Enter a value of 0 for allowance	*fAllowance = 0	Goes back to main menu.	Goes back to main menu.	Pass
	3	Enter a value for allowance that’s less than 13000	*fAllowance = 12997	“MINIMUM ALLOWANCE MUST BE 13000!”, *fAllowance = 0, Then prompts for another input.	“MINIMUM ALLOWANCE MUST BE 13000!”, *fAllowance = 0, Then prompts for another input.	Pass
	4	Enter an allowance more than or equal to 13000, then confirm your input	*fAllowance = 15000, *cInput = ‘1’	*fAllowance = 15000, then goes back to main menu or the next function	*fAllowance = 15000, then goes back to main menu or the next function	Pass
	5	Enter an allowance more than or equal to 13000, but don’t confirm your input	*fAllowance = 13000, *cInput = ‘0’	*fAllowance = 0	*fAllowance = 0	Pass
allocBudget	1	Select Food and Drinks, then enter a value less than 300	nInput = 1, *fFoodB = 299	“MINIMUM BUDGET FOR FOOD AND DRINKS IS 300!”, *fFood = 300, then prompts for another input	“MINIMUM BUDGET FOR FOOD AND DRINKS IS 300!”, *fFood = 300, then prompts for another input	Pass
	2	Select Housing and Utilities, then enter a value less than 4000	nInput = 4, *fHouseB = 2000	“MINIMUM BUDGET FOR HOUSING AND UTILITIES IS 4000!”, *fHouseB = 4000, then prompts for another input	“MINIMUM BUDGET FOR HOUSING AND UTILITIES IS 4000!”, *fHouseB = 4000, then prompts for another input	Pass
	3	Select Food and Drinks, then enter a value more than or equal to 300	nInput = 1, *fFoodB = 450	*fFoodB = 450, then goes back to Budget Category Menu	*fFoodB = 450, then goes back to Budget Category Menu	Pass
	4	Select Housing and Utilities, then enter a value more than or equal to 4000	nInput = 4 *fHouseB = 4001	*fHouseB = 4001, then goes back to Budget Category Menu	*fHouseB = 4001, then goes back to Budget Category Menu	Pass
	5	Select Housing and Utilities/Food and Drinks, then enter 0	nInput = 4, *fHouseB = 0	*fHouseB = 4000, then goes back to Budget Category Menu	*fHouseB = 4000, then goes back to Budget Category Menu	Pass
	6	Select any category, enter an appropriate non-negative number, then Finish Budget, and Confirm your allocation	nInput = 2, *fTransB = 550, nInput = 0, *cInput = ‘1’	*fTransB = 550, then goes back to main menu or the next function	*fTransB = 550, then goes back to main menu or the next function	Pass
	7	Select Healthcare, Enter Appropriate Budget Amount (i.e., non-negative), then Finish Budget, but do not Confirm your allocation	nInput = 3, *fHealthB = 1000, nInput = 0, *cInput = ‘X’	*fHealthB = 0, then goes back to main menu or the next function	*fHealthB = 0, then goes back to main menu or the next function	Pass
	8	Enter budget amount that exceeds user’s current balance of 15000	*fBalance = 15000, nInput = 3, *fHealthB = 1500, nInput = 5, *fLeisureB = 1500, nInput = 0, *cInput = 1	*fBalance = 15000, *fHealthB = 1500, *fLeisure = 1500, *fFoodB * 30 = 9000, *fHouse = 4000, *fTotalBudget = 16000, “TOTAL BUDGET ALLOCATION MUST NOT BE GREATER THAN YOUR CURRENT BALANCE!”, budget values reset to 0 and user is prompted to enter budget again.	*fBalance = 15000, *fHealthB = 1500, *fLeisure = 1500, *fFoodB * 30 = 9000, *fHouse = 4000, *fTotalBudget = 16000, “TOTAL BUDGET ALLOCATION MUST NOT BE GREATER THAN YOUR CURRENT BALANCE!”, budget values and user is prompted to enter budget again.	Pass
allocExpense	1	Enter a nonnegative amount for Expense that’s less than or equal to Balance	nInput = 1, *fFood = 50, nInput = 0	*fBalance = 13000 *fFood = 50 *fExpenses = 50 *fBalance = 12950	*fBalance = 13000 *fFood = 50 *fExpenses = 50 *fBalance = 12950	Pass

	2	Enter a negative amount for Expense	nInput = 1, *fTrans = -29	“Invalid Input”, then goes back to Expense Category Menu. *fTrans = 0	“Invalid Input”, then goes back to Expense Category Menu. *fTrans = 0	Pass
	3	Enter an option that’s more than 6 or less than 0	nInput = 10,	“Invalid Input.”, then prompts again for another input.	“Invalid Input.”, then prompts again for another input.	Pass
	4	Enter a nonnegative amount for Expense that’s more than Balance (e.g., 13000)	nInput = 5, *fLeisure = 13001	“Your expenses have gone beyond your current balance.”, then prompts for new input again.	“Your expenses have gone beyond your current balance.”, then prompts for new input again.	Pass
addSavings	1	Enter a negative value	fAddSavings = -300	“Invalid Input.” Then goes back to user actions menu, *fSavings = 0	“Invalid Input.” Then goes back to user actions menu, *fSavings = 0	Pass
	2	Enter a positive value that’s less than or equal to the balance	*fBalance = 13000 fAddSavings = 4250.99	*fSavings = 4250.99 *fSavings30 = 4250.99 *fBalance = 10749.01	*fSavings = 4250.99 *fSavings30 = 4250.99 *fBalance = 10749.01	Pass
	3	Enter a positive value that’s more than the balance	*fBalance = 15000 fAddSavings = 15000.1	“ADDED SAVINGS MUST NOT BE MORE THAN YOUR CURRENT BALANCE!”, then prompts again for new input	“ADDED SAVINGS MUST NOT BE MORE THAN YOUR CURRENT BALANCE!”, then prompts again for new input	Pass
useSavings	1	Enter a negative value	*fUseSavings = -100	“Invalid Input”, then prompts again for new input	“Invalid Input”, then prompts again for new input	Pass
	2	Enter a value that’s less than or equal to current Total Savings, but less than the current month’s savings	*fBalance = 13000, *fSavings = 2000, *fSavings30 = 1000, *fUseSavings = 999	*fSavings = 1001, *fSavings30 = 1, *fBalance = 13999	*fSavings = 1001, *fSavings30 = 1, *fBalance = 13999	Pass
	3	Enter a value that’s less than or equal to current Total Savings, but more than or equal to current month’s savings	*fBalance = 15000, *fSavings = 1500, *fSavings30 = 500, *fUseSavings = 700	*fSavings = 800, *fSavings30 = 0, *fBalance = 15700	*fSavings = 800, *fSavings30 = 0, *fBalance = 15700	Pass
	4	Enter a value that’s more than current Total Savings	*fBalance = 15000, *fSavings = 2000, *fUseSavings = 2500	“!SAVINGS THAT WILL BE USED MUST BE EQUAL TO TOTAL SAVINGS AT MOST!”, then prompts again for new input, *fSavings = 2000, *fBalance = 15000	“!SAVINGS THAT WILL BE USED MUST BE EQUAL TO TOTAL SAVINGS AT MOST!”, then prompts again for new input, *fSavings = 2000, *fBalance = 15000	Pass
endDay	1	Don’t confirm ending the day (Enter any key except ‘1’)	*nDay = 21 cInput = ‘x’	*nDay = 21, goes back to user actions menu	*nDay = 21, goes back to user actions menu	Pass
	2	Confirm ending the day when it is not the end of the month	*fBalance = 15000, *fExpenses = 0, *fFood = 0, *nDay = 1, cInput = ‘1’	*fBalance = 14700, *fExpenses = 300, *fFood = 300, *nDay = 2,	*fBalance = 14700, *fExpenses = 300, *fFood = 300, *nDay = 2,	Pass
	3	Confirm ending the day when it is the end of the month (day 30, 60, 90…)	*fBalance = 6300 *fAllowance = 14000 *fExpenses = 8700 *nDay = 60 *nMonth = 2 cInput = ‘1’	*fFood = 9000, *fHouse = 4000, *fExpenses = 13000, *fBalance = 2000, *nDay = 61, *nMonth = 3, *fBalance = *fBalance + *fAllowance, *fBalance = 16000	*fFood = 9000, *fHouse = 4000, *fExpenses = 13000, *fBalance = 2000, *nDay = 61, *nMonth = 3, *fBalance = *fBalance + *fAllowance, *fBalance = 16000	Pass
	4	Confirm ending the day when your balance is less than 300 when it’s not the end of the month	*fBalance = 250 *nDay = 21 cInput = ‘1’	*nDay = 22 *fBalance = -50 “ALERT! YOU ARE BROKE!” then the program gets terminated	*nDay = 22 *fBalance = -50 “ALERT! YOU ARE BROKE!” then the program gets terminated	Pass
	5	Confirm ending the day when your balance is less than 4300 at the end of the month	*fBalance = 4000 *nDay = 30 *fHouse = 0 *fFood = 8700 *fExpenses = 8700 cInput = ‘1’	*nDay = 31 *fHouse = 4000 *fFood = 9000 *fExpenses = 13000 *fBalance = -300	*nDay = 31 *fHouse = 4000 *fFood = 9000 *fExpenses = 13000 *fBalance = -300	Pass
endMonth	1	Don’t confirm ending the month (Enter any key except ‘1’)	cInput = ‘x’ *nDay = 88	*nDay = 88, then goes back to user actions menu	*nDay = 88, then goes back to user actions menu	Pass

	2	Confirm ending the month when it’s not the end of the month	*nDay = 42, *nDay30 = 18, *fBalance = 9700, *fFood = 3300, *fExpenses = 3300, cInput = ‘1’	“You have skipped 18 days.” *nDay = 60, *fBalance = 9700 – 300*nDay30 = 4300, *fFood = 8700, *fExpenses = 8700	*nDay = 42 + 18 = 60, *fBalance = 9700 – 300*nDay30 = 4300, *fFood = 8700, *fExpenses = 8700	Pass
	3	Select to end the month when it’s the end of the month	*nDay = 90	"The only available option is to end the day.", *nDay = 90, then goes back to user actions menu.	"The only available option is to end the day.", *nDay = 90, then goes back to user actions menu.	Pass
	4	Confirm ending the month when your daily food spendings will be more than your balance.	nDay = 1, nDay30 = 29, *fBalance = 4700, *fExpenses = 2300, *fFood = 0, cInput = ‘1’	“You have skipped 29 days.” nDay = 30, *fFood = 8700, *fExpenses = 11000, *fBalance = -4000, “ALERT! YOU ARE BROKE!” then the program gets terminated	“You have skipped 29 days.” nDay = 30, *fFood = 8700, *fExpenses = 11000, *fBalance = -4000, “ALERT! YOU ARE BROKE!” then the program gets terminated	Pass
startBudget	1	Enter an uppercase X	cInput = ‘X’	Goes back to Main Menu	Goes back to Main Menu	Pass
	2	Enter a lowercase x	cInput = ‘x’	Goes back to Main Menu	Goes back to Main Menu	Pass
	3	Enter an uppercase A	cInput = ‘A’	Selects case a (Add Expenses)	Selects case a (Add Expenses)	Pass
	4	Enter a character that’s not a,b,c,d,e,x (uppercase or lowercase)	cInput = ‘1’	“Invalid Input.”, then prompts for another input.	“Invalid Input.”, then prompts for another input.	Pass