Test Cases (Make Me Fit v1.0)

Harvey Julian P. Ermitanio

Function Name	#	Test Description	Sample Input	Expected Result	Actual Result	P/F
FTtoInches	**1**	Input is 0 feet	nHeightFT = 0	0	0	**P**
		Input is a negative value	nHeightFT = -5	Returns 0 and prints “nHeightFT cannot be negative!”	Returns 0 and prints “nHeightFT cannot be negative!”	**P**
		Input is the base value (1)	nHeightFT = 1	12	12	**P**
		Input is small	nHeightFT = 5	60	60	**P**
		Input is huge	nHeightFT = 505	6060	6060	**P**
LbsToKG	**2**	Input is 0 lbs	fLbs = 0.000000	0. 000000	0. 000000	**P**
		Input is a negative value	fLbs = -2. 000000	Returns 0.000000 and prints “fLbs cannot be negative!”	Returns 0.000000 and prints “fLbs cannot be negative!”	**P**

		Input is the base value (1)	fLbs = 1. 000000	0.453592	0.453592	**P**
		Input is small	fLbs = 67. 000000	30.390663	30.390663	**P**
		Input is huge	fLbs = 9999. 000000	4535.466309	4535.466309	**P**
displayHeader	**3**	-	**-**	Displays “Make Me Fit” header	Displays “Make Me Fit” header	**P**
displayLineSeparator	**4**	nLength is 0 for dashes	nLength = 0 nType = 2	(None)	(None)	**P**
		nLength is 0 for equal sign	nLength = 0 nType = 1	(None)	(None)	**P**
		Both inputs are valid	nLength = 20 nType = 2	--------------------	--------------------	**P**
		Both inputs are valid	nLength = 20 nType = 1	====================	====================	**P**
		nLength is negative for dashes	nLength = -1 nType = 2	Length cannot be negative. Check displayLineSeparator() function.	Length cannot be negative. Check displayLineSeparator() function.	**P**
		nLength is negative for equal sign	nLength = -2 nType = 1	Length cannot be negative. Check displayLineSeparator() function.	Length cannot be negative. Check displayLineSeparator() function.	**P**
		nType is a non-type	nLength = -20 nType = 3	No such type. Please check displayLineSeparator() function	No such type. Please check displayLineSeparator() function	**P**

displaySubHeader	**5**	Display a valid title	nTitle = 2	Settings ===================================	Settings ===================================	**P**
		Display a non-existing title	nTitle = 6	No such title. Please check displaySubHeader() function.	No such title. Please check displaySubHeader() function.	**P**
displayExerciseList	**6**	Normal list with mixed selected/unselected items	arrActivities = {“Yoga”, 2.3, 1}, {“Golf”, 4.8, 1}, {“Zumba”, 7.8, 0} nSize = 3	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [SELECTED] [2] Golf (4.8 MET) [SELECTED] [3] Zumba (7.8 MET) [NOT SELECTED] ---------------------------------	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [SELECTED] [2] Golf (4.8 MET) [SELECTED] [3] Zumba (7.8 MET) [NOT SELECTED] ---------------------------------	**P**
		All activities selected	arrActivities = {“Yoga”, 2.3, 1}, {“Golf”, 4.8, 1}, {“Zumba”, 7.8, 1} nSize = 3	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [SELECTED] [2] Golf (4.8 MET) [SELECTED] [3] Zumba (7.8 MET) [SELECTED] ---------------------------------	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [SELECTED] [2] Golf (4.8 MET) [SELECTED] [3] Zumba (7.8 MET) [SELECTED] ---------------------------------	**P**
		All activities NOT selected	arrActivities = {“Yoga”, 2.3, 0}, {“Golf”, 4.8, 0}, {“Zumba”, 7.8, 10 nSize = 3	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [NOT SELECTED] [2] Golf (4.8 MET) [NOT SELECTED] [3] Zumba (7.8 MET) [NOT SELECTED] ---------------------------------	=================================== Exercise List =================================== [1] Yoga (2.3 MET) [NOT SELECTED] [2] Golf (4.8 MET) [NOT SELECTED] [3] Zumba (7.8 MET) [NOT SELECTED] ---------------------------------	**P**
		Empty array (all names blank)	arrActivities = {“”, 0, 0}	=================================== Exercise List ===================================	=================================== Exercise List ===================================	**P**

			nSize = 1	---------------------------------	---------------------------------	
		List with gaps in between	arrActivities = {"Hiking", 6.0, 1}, {"", 0.0, 0}, {"Skip", 12.0, 1}, {"", 0.0, 0}, {"Walking", 3.0, 0}	=================================== Exercise List =================================== [1] Hiking (6.0 MET) [SELECTED] [2] Skip (12.0 MET) [SELECTED] [3] Walking (3.0 MET) [NOT SELECTED] ---------------------------------	=================================== Exercise List =================================== [1] Hiking (6.0 MET) [SELECTED] [2] Skip (12.0 MET) [SELECTED] [3] Walking (3.0 MET) [NOT SELECTED] ---------------------------------	**P**
askMenuOptions	**7**	Display a valid screen type	nScreenType = 1	[1] Start recommendation [2] Settings [3] Exit Enter option:	[1] Start recommendation [2] Settings [3] Exit Enter option:	**P**
		Display a non-valid screen type	nScreenType = 0	No such screen type. Check askMenuOptions function.	No such screen type. Check askMenuOptions function.	**P**
validateMenuOption	**8**	Validate a valid input	nUserInput = 1 nScreenType = 1	Returns 1	Returns 1	**P**
		Validate an invalid input	nUserInput = 9 nScreenType = 2	Returns 0 and prints “Invalid input. Please try again.”	Returns 0 and prints “Invalid input. Please try again.”	**P**
		Validate an input with invalid screen type	nUserInput = 9 nScreenType = 3	Returns 0 and prints “No such screen type. Please check validateMenuOption function.”	Returns 0 and prints “No such screen type. Please check validateMenuOption function.”	**P**
getUserData	**9**	Input is a valid gender	cGender = ‘m’	Accepts input to cGender	Accepts input to cGender	**P**
		Input is an invalid gender	cGender = ‘z’	Invalid input. Please enter 'M' or 'F'	Invalid input. Please enter 'M' or 'F'	**P**

		Input is a valid weight	fWeight = 120	Accepts input to fWeightLbs	Accepts input to fWeightLbs	**P**
		Input is an invalid weight	fWeight = 0	Invalid input. Weight must be greater than 0.	Invalid input. Weight must be greater than 0.	**P**
		Input is a valid height (ft)	fHeightFt = 5	Accepts input to fHeightFt	Accepts input to fHeightFt	**P**
		Input is an invalid height (ft)	**f**HeightFt = -3	Invalid input. Height in feet cannot be negative.	Invalid input. Height in feet cannot be negative.	**P**
		Input is a valid height (in)	fHeightIn = 11	Accepts input to fHeightIn	Accepts input to fHeightIn	**P**
		Input is an invalid height (in)	fHeightIn = 12	Invalid input. Inches must be between 0 and 11.	Invalid input. Inches must be between 0 and 11.	**P**
calculateCaloriesBurned	**10**	Inputs are valid	nMinutes = 60 nWeightKg = 70 fMET = 10	700	700	**P**
		One input is zero (MET)	nMinutes = 60 nWeightKg = 70 fMET = 0	0	0	**P**
		One input is negative (nMinutes)	nMinutes = -5 nWeightKg = 70 fMET = 10	0	0	**P**
		Input with long duration	nMinutes = 1440 nWeightKg = 70 fMET = 10	16800	16800	**P**
		Input with high MET	nMinutes = 20 nWeightKg = 70 fMET = 900	21,000	21,000	**P**

		Input with large weight	nMinutes = 20 nWeightKg = 200 fMET =10	2000	2000	**P**
computeBMI	**11**	Normal inputs	fWeight = 150 lbs nHeightFt = 5 nHeightIn = 6	24.207989	24.207989	**P**
		Input with zero value	fWeight = 0 nHeightFt = 5 nHeightIn = 6	0.000000	0.000000	**P**
		Input with negative value	fWeight = 150 nHeightFt = -5 nHeightIn = 6	nHeightFT cannot be negative!	nHeightFT cannot be negative!	**P**
		Extremely high weight	fWeight = 500 nHeightFt = 6 nHeightIn = 0	67.804787	67.804787	**P**
		Extremely tall height	fWeight = 150 nHeightFt = 8 nHeightIn = 0	11.442058	11.442058	**P**
		Minimum positive weight	fWeight = 1 nHeightFt = 5 nHeightIn = 6	0.161387	0.161387	**P**
		Minimum positive height	fWeight = 150 nHeightFt = 0 nHeightIn = 1	105450.000000	105450.000000	**P**
categorizeBMI		BMI is in underweight range	fBMI = 16.0	UNDERWEIGHT (0)	UNDERWEIGHT (0)	**P**

		BMI is in normal range (boundary)	fBMI = 18.5	NORMAL (1)	NORMAL (1)	**P**
		BMI is in overweight range (boundary)	fBMI = 25.0	OVERWEIGHT (2)	OVERWEIGHT (2)	**P**
		BMI is in obese range	fBMI = 30.0	OBESE (3)	OBESE (3)	**P**
		BMI is 0	fBMI = 0	UNDERWEIGHT (0)	UNDERWEIGHT (0)	**P**
calculateIdealWeight		Male, height below 5ft	nHeightFt = 4, nHeightIn = 10, cGender = 'M'	100	100	**P**
		Male, height exactly 5 ft	nHeightFt = 5, nHeightIn = 0, cGender = 'M'	105	105	**P**
		Male, height above 5 ft	nHeightFt = 5, nHeightIn = 8, cGender = 'M'	145	145	**P**
		Female, height below 5 ft	nHeightFt = 4, nHeightIn = 11, cGender = 'F'	91	91	**P**
		Female, height exactly 5 ft	nHeightFt = 5, nHeightIn = 0, cGender = 'F'	95	95	**P**
		Female, height above 5 ft	nHeightFt = 5, nHeightIn = 6, cGender = 'F'	119	119	**P**

		Edge case: 0 height	nHeightFt = 0, nHeightIn = 0, cGender = 'M'	0	0	**P**
		Very short height that causes negative ideal weight	nHeightFt = 2, nHeightIn = 0, cGender = 'F'	0	0	**P**
sortActsByMET		Mixed MET values	{ {"Yoga", 2.3, 1}, {"Zumba", 7.8, 0}, {"Golf", 4.8, 1}, {"HIIT", 9.0, 1}, {"Jog", 5.5, 0} }	{ {"HIIT", 9.0, 1}, {"Zumba", 7.8, 0}, {"Jog", 5.5, 0}, {"Golf", 4.8, 1}, {"Yoga", 2.3, 1} }	{ {"HIIT", 9.0, 1}, {"Zumba", 7.8, 0}, {"Jog", 5.5, 0}, {"Golf", 4.8, 1}, {"Yoga", 2.3, 1} }	**P**
		All same MET	{ {"Yoga", 5.0, 1}, {"Golf", 5.0, 0}, {"Jog", 5.0, 1} }	{ {"Yoga", 5.0, 1}, {"Golf", 5.0, 0}, {"Jog", 5.0, 1} }	{ {"Yoga", 5.0, 1}, {"Golf", 5.0, 0}, {"Jog", 5.0, 1} }	**P**
		Single Element	{ {"Yoga", 2.3, 1} }	{ {"Yoga", 2.3, 1} }	{ {"Yoga", 2.3, 1} }	**P**
		Empty Array	{}	{}	{}	**P**
allocateMinutesPerExercise		Only one selected activity	{"Yoga", 2.5, 1}, {"Zumba", 7.8, 0}, {"Jog", 5.5, 0} }, fWeightLbs=180, fIdealWeight=170	nFirstActMinutes=90, fFirstActCalBurned = 204.5, nSecondActIndex=-1, fSecondActMinutes=0, fSecondActCalBurned=0	nFirstActMinutes=90, fFirstActCalBurned = 204.5, nSecondActIndex=-1, fSecondActMinutes=0, fSecondActCalBurned=0	**P**

		Multiple selected activities	{ {“Yoga”, 2.5, 1}, {“Zumba”, 7.8, 1}, {“Jog”, 5.5, 1} }, fWeightLbs=150, fIdealWeight=140	nFirstActMinutes=90, fFirstActCalBurned = 204.3, nSecondActIndex=1 (Zumba), fSecondActMinutes=7, fSecondActCalBurned = 14.3	nFirstActMinutes=90, fFirstActCalBurned = 204.3, nSecondActIndex=1 (Zumba), fSecondActMinutes=7, fSecondActCalBurned = 14.3	**P**
displayResults		Category is NORMAL	arrActivities = {{"Walking", 4.0, 1}, {"Yoga", 2.5, 1}}, fWeightLbs = 160, fIdealWeight = 160, nCategory = NORMAL	“Keep up what you're doing. Don't forget to eat healthy foods while exercising!"	“Keep up what you're doing. Don't forget to eat healthy foods while exercising!"	**P**
		Category is UNDERWEIGHT	arrActivities = {{"Brisk Walking", 4.3, 1}, {"Swimming", 7.0, 1}}, fWeightLbs = 100, fIdealWeight = 120, nCategory = UNDERWEIGHT	Prints ideal weight, calls recommendProgramUnderweight, displays exercise suggestions	"Your ideal weight is: 120.00 lbs [Brisk Walking, Swimming, Golf, Weight Lifting] suggested by recommendProgramUnderweight"	**P**
		Category is OVERWEIGHT	arrActivities = {{"Running", 12.0, 1}, {"Walking", 4.0,	Assigns minutes for first and second activity, prints minutes, calories, total	[1] Running (12.00 MET)\ Minutes per day: 90 Calories burned: 2160.00	**P**

			1}, {"Yoga", 2.5, 1}}, fWeightLbs = 200, fIdealWeight = 180, nCategory = OVERWEIGHT	program duration, total daily minutes and calories burned	[2] Walking (4.00 MET) Minutes per day: 0 Calories burned: 0.00 Total program duration: 0.46 weeks Total daily minutes: 90 Total daily calories burned: 2160.00	
		OBESE: significantly overweight user	arrActivities = {{"Running", 12.0, 1}, {"Swimming", 7.0, 1}, {"Walking", 4.0, 1}, {"Yoga", 2.5, 1}}, fWeightLbs = 250, fIdealWeight = 200, nCategory = OBESE	Same as OVERWEIGHT but higher total weeks and daily calories burned; first and second activities allocated	[1] Running (12.00 MET) Minutes per day: 90 Calories burned: 2700.00 [2] Swimming (7.00 MET) Minutes per day: 0 Calories burned: 0.00 Total program duration: 0.65 weeks Total daily minutes: 90 Total daily calories burned: 2700.00	**P**