Notes:
- The *italicized parts* are "outputs" where a value just gets stored in a variable
  (This was tested in a different file by printing what's inside those variables after the function call)
- The **bold parts** are those displayed in the terminal
- The rest are descriptions of the outcomes

Function Name	#	Test Description	Sample Input	Expected Result	Actual Result	P/F
main()	1	maintenance code is inputted in getCrust	see getCrust()_2	**Crust: 80808** -- **Machine will be shut down for MAINTENANCE.** -- exits the program	**Crust: 80808** -- **Machine will be shut down for MAINTENANCE.** -- exits the program	P
	2	order is cancelled at getSize()	see getSize()_3	**Size: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	**Size: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	P
	3	order is cancelled at getToppings()	see getToppings()_4 //PART2	**Topping # 1: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	**Topping # 1: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	P
	4	order is cancelled at getCheese()	see getCheese()_3	**Cheese # 1: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	**Cheese # 1: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	P
	5	order is cancelled at getStuffings()	see getStuffing()_3	**Stuffing: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	**Stuffing: x** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	P
	6	order is cancelled at payment()	see payment()_4	**Pay: 0** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	**Pay: 0** -- **Order has been CANCELLED** -- nInvoice stays the same, back to step 1	P
	7	successful transactions until tasked to go under maintenance	2 orders then tasked to go under maintenance	-- **Welcome to Easy Pizza!** [...steps...] **Invoice Number: 0001** (saved to Receipts.txt) [...steps...] **We hope you order again soon!** -- **Welcome to Easy Pizza!** [...steps...] **Invoice Number: 0002** (saved to Receipts.txt) [...steps...] **We hope you order again soon!** -- **Welcome to Easy Pizza!** [...getCrust()...] **Crust: 80808** -- **Machine will be shut down for MAINTENANCE.** -- exits the program	-- **Welcome to Easy Pizza!** [...steps...] **Invoice Number: 0001** (saved to Receipts.txt) [...steps...] **We hope you order again soon!** -- **Welcome to Easy Pizza!** [...steps...] **Invoice Number: 0002** (saved to Receipts.txt) [...steps...] **We hope you order again soon!** -- **Welcome to Easy Pizza!** [...getCrust()...] **Crust: 80808** -- **Machine will be shut down for MAINTENANCE.** -- exits the program	P
getCrust()	1	input is valid (1 or 2)	2	_*pCrust = 2_	_*pCrust = 2_	P
	2	input is 80808	80808	_*pMaintenance = 1_	_*pMaintenance = 1_	P
	3	input is invalid	234	**Invalid Input. Choose a valid crust.** and asks to input again until input is valid	**Invalid Input. Choose a valid crust.** and asks to input again until input is valid	P
getSize()	1	input is valid ('s', 'm', 'l', 'S', 'M', 'L'), nCrust = 1	'M' (nCrust = 1)	**TOTAL 175.00** _*pSize = M_ _crust_size[1].nQty = 1_	**TOTAL 175.00** _*pSize = M_ _crust_size[1].nQty = 1_	P
	2	input is valid ('s', 'm', 'l', 'S', 'M', 'L'), nCrust = 2	's' (nCrust = 2)	**TOTAL 100.00** _*pSize = S_ _crust_size[3].nQty = 1_	**TOTAL 100.00** _*pSize = S_ _crust_size[3].nQty = 1_	P
	3	input is 'x' or 'X'	'X'	_*pCancel = 1_	_*pCancel = 1_	P
	4	input is invalid	'G'	**Invalid Input. Choose a valid size.** and asks to input again until input is valid	**Invalid Input. Choose a valid size.** and asks to input again until input is valid	P
getToppings() // PART 1	1	size ordered is small	cSize = 'S'	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 10.00)** **[P] - Pineapple (PhP 6.00)** **[S] - Sausage (PhP 15.00)** **[C] - Cheese (PhP 12.00)** **[O] - Olives (PhP 10.00)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 10.00)** **[P] - Pineapple (PhP 6.00)** **[S] - Sausage (PhP 15.00)** **[C] - Cheese (PhP 12.00)** **[O] - Olives (PhP 10.00)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	P

	2	size ordered is medium	cSize = 'M'	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 15.00)** **[P] - Pineapple (PhP 9.00)** **[S] - Sausage (PhP 22.50)** **[C] - Cheese (PhP 18.00)** **[O] - Olives (PhP 15.00)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 15.00)** **[P] - Pineapple (PhP 9.00)** **[S] - Sausage (PhP 22.50)** **[C] - Cheese (PhP 18.00)** **[O] - Olives (PhP 15.00)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	P
	3	size ordered is large	cSize = 'L'	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 17.50)** **[P] - Pineapple (PhP 10.50)** **[S] - Sausage (PhP 26.25)** **[C] - Cheese (PhP 21.00)** **[O] - Olives (PhP 17.50)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	**|STEP 3|** **Choose at least 1 topping:** **[H] - Ham (PhP 17.50)** **[P] - Pineapple (PhP 10.50)** **[S] - Sausage (PhP 26.25)** **[C] - Cheese (PhP 21.00)** **[O] - Olives (PhP 17.50)** **[N] - NEXT STEP** **[X] - CANCEL ORDER**	P
getToppings() // PART 2	1	inputs at least 1 valid topping ('h', 'p', 's', 'c', 'o', 'H', 'P', 'S', 'C', 'O')	'h', 'n' (cSize = 'S')	**Topping # 1:** h **TOTAL: 110.00** **Topping # 2:** n *toppings[0].nQty = 1*	**Topping # 1:** h **TOTAL: 110.00** **Topping # 2:** n *toppings[0].nQty = 1*	P
	2	inputs an invalid character	h', 'z', 'c', 'z', 'n' (cSize = 'M')	**Topping # 1:** h **TOTAL: 190.00** **Topping # 2:** z **Invalid Input. Choose a valid topping.** **Topping # 2:** c **TOTAL: 208.00** **Topping # 3:** z **Invalid Input. Choose a valid topping.** **Topping # 3:** c **TOTAL: 226.00** **Topping # 4:** n *toppings[0].nQty = 1, toppings[3].nQty = 2*	**Topping # 1:** h **TOTAL: 190.00** **Topping # 2:** z **Invalid Input. Choose a valid topping.** **Topping # 2:** c **TOTAL: 208.00** **Topping # 3:** z **Invalid Input. Choose a valid topping.** **Topping # 3:** c **TOTAL: 226.00** **Topping # 4:** n *toppings[0].nQty = 1, toppings[3].nQty = 2*	P
	3	inputs 'n' or 'N' without having chosen toppings	'n'	**You must select at least 1 topping.** **Topping # 1:** (asks user to input again until at least 1 topping is selected)	**You must select at least 1 topping.** **Topping # 1:** (asks user to input again until at least 1 topping is selected)	P
	4	input is 'x' or 'X'	'X'	**pCancel = 1*	**pCancel = 1*	P
getCheese()	1	all inputs are valid ('m', 'g', 'f', 'p', 'M', 'G', 'F', 'P')	'm', 'g' (nQty = 2, fTotal = 124.0)	**Cheese # 1:** m **Cheese # 2:** g **TOTAL: 124.00** *cheeses[0].nQty = 1, cheeses[1].nQty = 1*	**Cheese # 1:** m **Cheese # 2:** g **TOTAL: 124.00** *cheeses[0].nQty = 1, cheeses[1].nQty = 1*	P
	2	inputs an invalid character	'p', 'a', 'f' (nQty = 2, fTotal = 124.0)	**Cheese # 1:** p **Cheese # 2:** a **Invalid Input. Choose a valid cheese type.** **Cheese # 2:** f **TOTAL: 124.00** *cheeses[2].nQty = 1, cheeses[3].nQty = 1*	**Cheese # 1:** p **Cheese # 2:** a **Invalid Input. Choose a valid cheese type.** **Cheese # 2:** f **TOTAL: 124.00** *cheeses[2].nQty = 1, cheeses[3].nQty = 1*	P
	3	inputs 'x' or 'X'	'm', 'x' (nQty = 2, fTotal = 124.0)	**Cheese # 1:** m **Cheese # 2:** x **pCancel = 1*	**Cheese # 1:** m **Cheese # 2:** x **pCancel = 1*	P
getStuffing()	1	input is valid stuffing ('c', 's', 'b', 'C', 'S', 'B')	'c' (*pTotal = 110.0)	**Stuffing:** c **TOTAL: 140.00** *stuffings[0].nQty = 1*	**Stuffing:** c **TOTAL: 140.00** *stuffings[0].nQty = 1*	P
	2	input is no stuffing ('n', 'N')	'N' (*pTotal = 110.0)	**Stuffing:** N **TOTAL: 110.00**	**Stuffing:** N **TOTAL: 110.00**	P
	3	input is 'x' or 'X'	'x' (*pTotal = 110.0)	**pCancel = 1*	**pCancel = 1*	P
	4	input is invalid	'm', 'b' (*pTotal = 110.0)	**Stuffing:** m **Invalid Input. Choose a valid stuffing.** **Stuffing:** b **TOTAL: 145.00** *stuffings[2].nQty = 1*	**Stuffing:** m **Invalid Input. Choose a valid stuffing.** **Stuffing:** b **TOTAL: 145.00** *stuffings[2].nQty = 1*	P
payment()	1	pays exact amount	(fTotal = 145.00) 3, 5, 5, 7	**Pay:** 3 **TOTAL: 45.00** **Pay:** 5 **TOTAL: 25.00** **Pay:** 5 **TOTAL: 5.00** **Pay:** 7 **TOTAL: 0.00**	**Pay:** 3 **TOTAL: 45.00** **Pay:** 5 **TOTAL: 25.00** **Pay:** 5 **TOTAL: 5.00** **Pay:** 7 **TOTAL: 0.00**	P
	2	pays amount greater than total	(fTotal = 242.50) 12, 11, 10, 9 ,2	**Pay:** 12 **TOTAL: 242.49** **Pay:** 11 **TOTAL: 242.44** **Pay:** 10 **TOTAL: 242.34** **Pay:** 9 **TOTAL: 242.09** **Pay:** 2 **TOTAL: -257.91** **Change:** **(2) 100 peso bills** **(1) 50 peso bill** **(1) 5 peso coin** **(2) 1 peso coins** **(3) 25 centavo coins** **(1) 10 centavo coin** **(1) 5 centavo coin** **(1) 1 centavo coin**	**Pay:** 12 **TOTAL: 242.49** **Pay:** 11 **TOTAL: 242.44** **Pay:** 10 **TOTAL: 242.34** **Pay:** 9 **TOTAL: 242.09** **Pay:** 2 **TOTAL: -257.91** **Change:** **(2) 100 peso bills** **(1) 50 peso bill** **(1) 5 peso coin** **(2) 1 peso coins** **(3) 25 centavo coins** **(1) 10 centavo coin** **(1) 5 centavo coin** **(1) 1 centavo coin**	
	3	inputs invalid key	(fTotal = 146.00) 13, 2	**Pay:** 13 **Invalid Input.** **Enter Keys from 0-12 only.** **Pay:** 2 **TOTAL: -354.00** **Change:** **(3) 100 peso bills** **(1) 50 peso bill** **(4) 1 peso coins**	**Pay:** 13 **Invalid Input.** **Enter Keys from 0-12 only.** **Pay:** 2 **TOTAL: -354.00** **Change:** **(3) 100 peso bills** **(1) 50 peso bill** **(4) 1 peso coins**	P
	4	cancels order	0	**pCancel = 1*	**pCancel = 1*	P
receipt()	1	no cheese or stuffings	crust_size.nQty = 1 toppings[0].nQty = 1 (rest is 0) fTotal = 110.00 nInvoice = 2	**Invoice Number: 0002** **ITEM QTY PRICE** **Thin Crust [S] 1 100.00** **Ham 1 10.00** **TOTAL: 110.00** this is also stored in the Receipts.txt file	**Invoice Number: 0002** **ITEM QTY PRICE** **Thin Crust [S] 1 100.00** **Ham 1 10.00** **TOTAL: 110.00** this is also stored in the Receipts.txt file	P

	2	with cheese	crust_size[1].nQty = 1 toppings[3].nQty = 2 toppings[4].nQty = 1 cheeses[0].nQty = 1 cheeses[1].nQty = 1 (rest is 0) fTotal = 226.00 nInvoice = 3	**Invoice Number: 0003** **ITEM QTY PRICE** **Thin Crust [M] 1 175.00** **Cheese 2 36.00** **Mozzarella x 1** **Gorgonzola x 1** **Olives 1 15.00** **TOTAL: 226.00** this is also stored in the Receipts.txt file	**Invoice Number: 0003** **ITEM QTY PRICE** **Thin Crust [M] 1 175.00** **Cheese 2 36.00** **Mozzarella x 1** **Gorgonzola x 1** **Olives 1 15.00** **TOTAL: 226.00** this is also stored in the Receipts.txt file	P
	3	with stuffings	crust_size[5].nQty = 1 toppings[0].nQty = 0 toppings[1].nQty = 1 toppings[2].nQty = 1 toppings[3].nQty = 4 toppings[4].nQty = 1 stuffings[2].nQty = 1 (rest is 0) fTotal = 415.75 nInvoice = 5	**Invoice Number: 0005** **ITEM QTY PRICE** **Thick Crust [L] 1 225.00** **Ham 1 17.50** **Pineapple 1 10.50** **Sausage 1 26.25** **Cheese 4 84.00** **Mozzarella x 1** **Gorgonzola x 2** **Parmesan x 1** **Olives 1 17.50** **Stuffing: Both C&S 1 35.00** **TOTAL: 415.75**	**Invoice Number: 0005** **ITEM QTY PRICE** **Thick Crust [L] 1 225.00** **Ham 1 17.50** **Pineapple 1 10.50** **Sausage 1 26.25** **Cheese 4 84.00** **Mozzarella x 1** **Gorgonzola x 2** **Parmesan x 1** **Olives 1 17.50** **Stuffing: Both C&S 1 35.00** **TOTAL: 415.75**	P
calcTime()	1	only 1 topping; no stuffings	toppings[0].nQty = 1	182	182	P
	2	multiple toppings, including cheese; no stuffings	toppings[3].nQty = 2 toppings[4].nQty = 1	186	186	P
	3	with stuffings	toppings[0].nQty = 0 toppings[1].nQty = 1 toppings[2].nQty = 1 toppings[3].nQty = 4 toppings[4].nQty = 1 stuffings[2].nQty = 1	206	206	P
creation()	1	nTime is not 6 mod 7	186	**3:06 3:05 3:04 3:03 3:02 3:01 3:00** **2:59 2:58 2:57 2:56 2:55 2:54 2:53** [...actual values...] **0:11 0:10 0:09 0:08 0:07 0:06 0:05** **0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	**3:06 3:05 3:04 3:03 3:02 3:01 3:00** **2:59 2:58 2:57 2:56 2:55 2:54 2:53** [...actual values...] **0:11 0:10 0:09 0:08 0:07 0:06 0:05** **0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	P
	2	nTime is 6 mod 7 (for next line screen design purposes)	188	**3:08 3:07 3:06 3:05 3:04 3:03 3:02** **3:01 3:00 2:59 2:58 2:57 2:56 2:55** [...actual values] **0:13 0:12 0:11 0:10 0:09 0:08 0:07** **0:06 0:05 0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	**3:08 3:07 3:06 3:05 3:04 3:03 3:02** **3:01 3:00 2:59 2:58 2:57 2:56 2:55** [...actual values] **0:13 0:12 0:11 0:10 0:09 0:08 0:07** **0:06 0:05 0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	P
	3	nTime is greater than 3 minutes	328	**5:28 5:27 5:26 5:25 5:24 5:23 5:22** **5:21 5:20 5:19 5:18 5:17 5:16 5:15** [...actual values...] **0:13 0:12 0:11 0:10 0:09 0:08 0:07** **0:06 0:05 0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	**5:28 5:27 5:26 5:25 5:24 5:23 5:22** **5:21 5:20 5:19 5:18 5:17 5:16 5:15** [...actual values...] **0:13 0:12 0:11 0:10 0:09 0:08 0:07** **0:06 0:05 0:04 0:03 0:02 0:01 0:00** **Thank you for waiting!**	P
toUpper()	1	input small letter character	'a'	'A'	'A'	P
	2	input capital letter character	'Y'	'Y'	'Y'	P
	3	input non-letter character	'!'	'!'	'!'	P