Author: Benjamin R. Olson
Date: April 24, 2014
Course: CS162 – Introduction to Computer Science II
Project 2 Test Plan

Items Class Function Tests

Procedure: Tests are hard-coded into test_items.cpp. First, item object created. Then, functions called on the object created, with results for each test conveyed in standard output.

Function #1: std::vector<Items::item> * Items::getAisle(int location)

Case	Function Call	Expected Return	Actual Return	Error(s) Found	Correction Made
Positive Case	getAisle(5)	std::vector<item> pasta	0	1. default in switch case, 2. never returns item_ptr	1. default: item_ptr = NULL, 2. item_ptr returned and re-tested for success
Negative Case	getAisle(-1)	NULL	0 (same as NULL)	none	na
Boundary Case	getAisle(0)	std::vector<item> produce	0x7fff9dd03cc0 (location of produce in memory)	none	na
Boundary Case	getAisle(9)	std::vector<item> paper	0x7fff9dd03d98 (location of produce in memory)	none	na
Negative Case	getAisle(10)	NULL	0 (same as NULL)	none	na

Function #2: std::string Items::getAisleName(int location)

Case	Function Call	Expected Return	Actual Return	Error(s) Found	Correction Made
Positive Case	getAisleName(5)	std::string “Pasta/Sauces/Etc.”	(same)	none	na
Negative Case	getAisleName(-1)	No return, program crashes	Seg. fault	none	na
Boundary Case	getAisleName(0)	std::string “Fresh Produce”	(same)	none	na
Boundary Case	getAisleName(9)	std::string “Paper”	(same)	none	na
Negative Case	getAisleName(10)	No return, program crashes	Seg. fault	none	na

Function #3: void Items::addItem(int location, std::string name, int quantity)

Case	Function Call	Expected Result	Actual Result	Error(s) Found	Correction Made
Positive Case	addItem(5, “noodles”, 2)	Struct item{name= “noodles”, quantity = 2} created in vector pasta	Program does not compile.	(i.getAisle(5))->item.at(0).name in function call	(*i.getAisle(5)).at(0).name and re-tested for success
Negative Case	addItem(-1, “reject”, 3)	No item added to any vector.	(same)	None	na
Boundary Case	addItem(0, “bananas”, 12)	Struct item{name= “bananas”, quantity = 12} created in vector pasta	(same)	None	na
Boundary Case	addItem(9, “toilet paper”, 6)	Struct item{name= “toilet paper”, quantity = 6} created in vector pasta	(same)	None	na
Negative Case	addItem(10, “reject2”, 3)	No item added to any vector.	(same)	None	na

Function #4: void Items::deleteItem(int location, int item_id)
Procedure: try to access an item after it has been deleted.

Note: Testing procedure for this funciton could be improved, yet the run-time error messages (in “Actual Results” are evidence that the function is working properly.

Case	Function Call	Expected Result	Actual Result	Error(s) Found	Correction Made
Positive Case	deleteItem(5, 0)	Some error (item doesn't exist because it has been erased).	Item exists unchanged (was not deleted as expected).	std::vector<item> i = *getAisle(location); in function deleteItem	item_ptr = getAisle(location); and re-tested for success: 'std::out_of_range' thrown – error as expected
Negative Case	deleteItem(-1, 0)	error	Seg. fault	none	na
Boundary Case	deleteItem(0, 0)	error	'std::out_of_range' thrown	none	na
Boundary Case	deleteItem(9, 0)	error	'std::out_of_range' thrown	none	na
Negative Case	deleteItem(10, 0)	error	Seg. fault	none	na

Input Validation Functions Tests:

**Note: During implementation, I realized that isAisleNumber and isValidQuantity could be combined into a single function, making input validation more modular and the code less redundant. So I strayed from the design, combining these two functions into the following function:

Function #5: bool isIntInRange(std::string input, int & i, int min, int max)
Procedure: Get string input from getline() in while loop and display pre- and post- conditions of function calls as well as the function return value.
Where args min = 1 and max = 10:

Positve Cases (should all return true and set int i to input): input = “1”, “5”, “10”
Negative Cases (should all return false and not set int I): input = “-1”, “11”, “x1”, “hello”
Results were as expected.

Where args min = 1 and max = 1000:

Positve Cases (should all return true and set int i to input): input = “1”, “500”, “1000”
Negative Cases (should all return false and not set int I): input = “-1”, “0”, “1001”, “x100”, “hello”, “00111”
I retested, fixing bugs, until results were as expected.

**Note: During implementation, I also realized that isValidItemName could be extended slightly to filter input for any maximum length, by introducing the formal parameter int max_chars.

Function #6: bool isValidItemName(std::string input, int max_chars)
Procedure: Get string input from getline() in while loop and display function return value.
Where max_chars = 20:

Positive Cases (should all return true): input = “hello”, “a”, “A”, “abcdefghijklmnopqrst”
Negative Cases (should all return false): input = \n, “abcdefghijklmnopqrstu”,
“abcdefghijklmnopqrstuv”

Remaining functions tested directly from running main.cpp:

void inputItemDeletion(int max_id, int & var)
void inputLocation(int & var)
void inputName(std::string & var)
void inputQuantity(int & var)
(other)