<u>Module:</u> getAllItemsInCostRange()

<u>Access Programs:</u> none

<u>Implementation:</u>

<u>Uses:</u> inputFileName.txt, setRemoveItem(), setItemData()

<u>Variables</u>

<u>Input:</u>
min_cost: LONG, max_cost: LONG
Represents the domain of the objects to be output with respect to their cost. Assumes min_cost < max_cost.

<u>Output:</u>
product: ARRAY(<STRING>,<BOOLEAN>)
Outputs a string containing data pertaining to a single object within the domain min_cost < (object cost) max_cost and a Boolean value indicating if there are more objects within the domain remaining in the data file

<u>State:</u>
object.used: BOOLEAN
Represents whether the object has already been previously used in a prior call

recorded_min_cost: LONG, recorded_max_cost: LONG
Represents current boundaries being used, input values that vary from these values trigger a reset to all object.used values to FALSE.

<u>Constants:</u> inputFileName: CHAR[]
Represents the name of the text file used by this module

<u>Psudo code:</u>

```
IF (min_cost == recorded_min_cost and  max_cost == recorded_max_cost) DO
    FOR (each object in data file) DO
        read the object
        IF ((min_cost <= (current_object.cost) <= max_cost) AND (object.used == FALSE)) DO
            IF (product[0] contains an object) DO
                product[1] = FALSE
                RETURN product
            ELSE DO
                product[1] = TRUE
            current_object.used = TRUE
            set product[0] to object
        END FOR
```

```
RETURN product
ELSE DO
        recorded_min_cost = min_cost;  recorded_max_cost =  max_cost
        FOR (each object in data file) DO
                object used = FALSE
        FOR (each object in data file) DO
                read the object
                IF ((min_cost <= (object cost) <= max_cost) AND (object_used == FALSE)) DO
                        IF (product[0] contains an object) DO
                                product[1] = FALSE
                                RETURN product
                        ELSE DO
                                product[1] = TRUE
                        object_used = TRUE
                        set product[0] to object
                END FOR
RETURN product
```

<u>Function table:</u>

		product[0]	product[1]
min_cost <= (current_object.cost) <= max_cost AND (current_object.used == FALSE	product[0] contains an object	NO CHANGE	FALSE
	product[0] does not contains an object	current_object	TRUE
ELSE		NO CHANGE	NO CHANGE
min_cost <= (current_object.cost) <= max_cost AND (current_object.used == FALSE	product[0] contains an object	NO CHANGE	FALSE
	product[0] does not contains an object	current_object	TRUE
ELSE		NO CHANGE	NO CHANGE

<u>Test Report:</u>

TEST CASE (object.code, object.cost, object.used	min_cost IN, max_cost IN	recorded_min_cost, recorded_max_cost	object.used (through iteration)	product OUT	Result
Item1, 3, FALSE Item2, 7, FALSE Item3, 5,	1, 9	0, 0	FALSE, FALSE, FALSE, FALSE, FALSE, FALSE	(Item1, FALSE)	pass

FALSE Item4, 50, FALSE Item5, 2, FALSE Item6, 10, FALSE	1, 9	1, 9	TRUE, FALSE, FALSE, FALSE, FALSE, FALSE	(Item2, FALSE)	pass
	1, 9	1, 9	TRUE, TRUE, FALSE, FALSE, FALSE, FALSE	(Item3,FALSE)	pass
	1, 9	1, 9	TRUE, TRUE, TRUE, FALSE, FALSE, FALSE	(Item5, TRUE)	pass
	1, 9	1, 9	TRUE, TRUE, TRUE, FALSE, TRUE, FALSE	error	Fail – exception when no possible objects remain not handled
	10, 100	1, 9	FALSE, FALSE, FALSE, FALSE, FALSE, FALSE	(Item4, FALSE)	pass