Machine Project

Test Script

Database Functions						
Function	**#**	**Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
countSymptoms	1	Base case	Symptoms[0] = Termination_ID	0	0	P
	2	Counting symptom array with 5 elements	Symptom[5] = Termination_ID	5	5	P
	3	Termination_ID not found within array	Symptoms	Infinite loop	Infinite loop	P
readSymptosId	1	Base case	Srclen = 0	Destlen = 0	Destlen = 0	P
	2	Reading a 5 elements	Srclen = 10	Destlen = 5	Destlen = 5	P
	3	Reading a string of size 10 but values are not space separated	Srclen = 10	Destlen = 1	Destlen = 1	P
readImpressionDB	1	Base case	Impressions.txt is empty	ERROR	ERROR	P
	2	Dbcount is 5	Impression.txt count = 5 … 5 impressions	Impressiondb.count = 5	Impressiodb.count = 5	P
	3	Dbcount does not match the number of impressions within the file	Impression.txt count = 10 … 2 impressions	SEGFAULT	SEGFAULT	P
getSymptoms	1	Base case	Initial empty database	Empty symptomarr	Empty symptomarr	P
	2	Db.count does not match number of symptoms in database	Db.count = 10 Number of symptoms = 2	SEGFAULT	SEGFAULT	P
	3	Storing 5 symptoms	Db.count = 5 … 5 symptoms	Symptomarrlen = 5	Symptomarrlen	P
getSymptomFromIDs	1	Base case, read first symptom	ID = 1	Return first symptom in db	Return first symptom in db	P
	2	Read last symptom	ID = 20	Return 20th symptom	Return 20th symptom	P
	3	Read nonexistent symptom	ID = -1	SEGFAULT	SEGFAULT	P
getImpressionFromName	1	Base case, read default name	Name = " "	Return first symptom	Return first symptom	P

	2	Case sensitive name	Name = “coVID”	“ “	“ “	P
	3	Find “Covid”	Name = “Covid”	Covid impressino struct	Covid Impressino struct	P

Utilities						
Function	**#**	**Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
isDigit	1	Base case, input c	c	0	0	P
	2	Input 1	1	1	1	P
	3	Input a new line	\n	0	0	P
toUpper	1	Base case, capitalize c	c	C	C	P
	2	Capitalize f	f	F	F	P
	3	Capitalize d	d	D	D	P
handleInt	1	Invalid case	abcd	Invalid input	Invalid input	P
	2	Base case, enter 1	1	1	1	P
	3	Enter c	c	Ascii val of c	Ascii val of c	P
handleCH	1	Base case, ref = “ABC”	C	C	C	P
	2	Invalid case, ref = “ABC”	D	Invalid input	Invalid input	P
	3	Ref is empty	Any input	Any input	Any input	P
handleStr	1	Base case, ref = “ABC”, input = “AB”	AB	AB	AB	P
	2	String is not a substring of ref	“DE”	Not in dataset, try again	Not in dataset, try again	P
	3	String input is larger than alotted	“ABCDEFGJILKMNEZP”	String overflow error	Sting overflow error	P
fileExists	1	Base case, filename = “Impressions.txt”	Impressions.txt	1	1	P
	2	Base case 2, filename = “Symptoms.txt”	Symptoms.txt	1	1	P
	3	Filename does not exist in the directory	"null.txt”	0	0	P
affirmative	1	Negative base case, question = “This selection is potentially destructive”	yes	0	0	P
	2	Positive base case	YES	1	1	P

	3	User types random string in response	abcde	0	0	P
isFound	1	User inputs ‘E’ for Pokemon type.	dex->collection [index].type = ‘E’	input = ‘E’	input = ‘E’	P
	2	User inputs ‘m’ for Pokemon type.	dex->collection [index].type = ‘m’	Invalid input. Please try again:	Invalid input. Please try again:	P
	3	User inputs ‘e’ for Pokemon type.	dex->collection [index].type = ‘e’	input = ‘e’	input = ‘e’	P

Note:

Initialization functions and wrappers were not included as unit testing them is largely impractical. Integration testing is a much more useful metric for measuring program cohesion in that respect.

Purely aestheitc functions were also excluded from the test script.