```
For this assignment, you will create a run(instruction) function that accepts an
array containing a function pointer and its parameters and calls the function
inside the array.  Your function should react according to the output of the
executed function.  You should be able to print a string (if the result looks
like it is in memory), print OK (result=0), and print an integer(all other cases).
Your function should also return 0 if the executed function returned 0 or a string,
and the integer in the other cases.  The length of the array will allow you to
determine the number of parameters that the function takes.
```

You should not need to modify the test suite. You only need to write your function. Since QT SPIM expects all of your code to be in a single file, you can concatenate them together in a few ways. If you are on Windows, you can use the included batch file to do the work for you. Simply dragging your source file and dropping it on the batch file should be sufficient. If you are having trouble with the batch file, make sure that your file names match those below. You can also use a command line operation.

Windows: copy /Y "<Your Source File Name>"+"Test Suite.asm" Output.asm

Unix: cat "<Your Source File Name>" "Test Suite.asm" > Output.asm

Your program should include appropriate comments indicating what the code should be doing and what registers are being used for. After displaying the results, your program should exit cleanly. You should test your programs using the SPIM simulator to ensure their functionality before submitting them. You should only submit your functions. You will not receive credit if you submit the test suite in any form. You should also not include any driver or debug code in your submission.

Objectives:

1. To introduce function pointers.
2. To practice with large parameter sets.
3. To review parameter checking.

Expected Output:

```
##-----------------------Starting functionality tests.-----------------------##
Hello world!
Test #1 passed:  Testing with the hello world function.
Two times 2 is 4.
OK
Test #2 passed:  Testing with the double function.
Two times 21 is 42.
OK
Test #3 passed:  Testing with the double function.
-1
Test #4 passed:  Testing with the f(0) function.
Happy, happy, joy, joy!
OK
Test #5 passed:  Testing with the f(3) function.
This is a string.
Test #6 passed:  Testing with the f(5) function.
3
Test #7 passed:  Testing with the max(1, 2, 3) function.
3
Test #8 passed:  Testing with the max(3, 2, 1) function.
3
Test #9 passed:  Testing with the max(1, 3, 2) function.
```

```
OK
Test #10 passed:  Testing with the print_sum() function.
1

1
Test #11 passed:  Testing with the print_sum(1) function.
1
2

3
Test #12 passed:  Testing with the print_sum(1, 2) function.
1
2
3

6
Test #13 passed:  Testing with the print_sum(1, 2, 3) function.
1
2
3
4

10
Test #14 passed:  Testing with the print_sum(1, 2, 3, 4) function.
1
2
3
4
5

15
Test #15 passed:  Testing with the print_sum(1, 2, 3, 4, 5) function.
1
2
3
4
5
6

21
Test #16 passed:  Testing with the print_sum(1, 2, 3, 4, 5, 6) function.
1
2
3
4
5
6
7
8
9
10
11
12
13

91
Test #17 passed:  Testing with the print_sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13) function.

##--------------------Starting parameter checking tests.--------------------##
```

```
Test #101 passed:  Null pointer check.
Test #102 passed:  Array pointer below zero.
Test #103 passed:  Pointer in text range.
Test #104 passed:  Pointer too high for dynamic data memory.
Test #105 passed:  Pointer is not word aligned.
Test #106 passed:  Pointer is not word aligned.
Test #107 passed:  Pointer is not word aligned.
Test #108 passed:  Negative array length.
Test #109 passed:  Array extends outside of memory.
Test #110 passed:  Array wraps around to the beginning of memory.
Test #111 passed:  Array is inside of the memory range, but attempts to cross
from the heap to the stack.
Test #112 passed:  Array is empty.
Test #113 passed:  Does not contain a function pointer.

##-----------------------------Testing completed.-----------------------------##
```