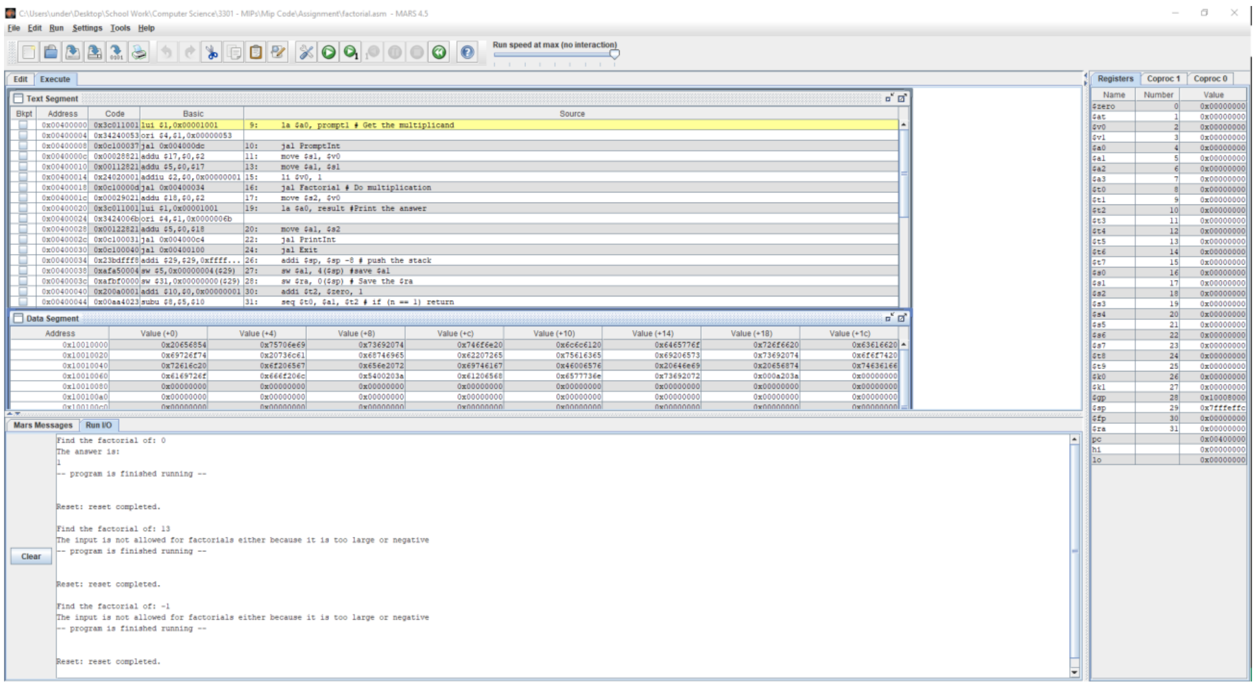
Part 1: Factorial Program


C:\Users\under\Desktop\School Work\Computer Science\3301 - MIPs\Mip Code\Assignment\factorial.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	9: la $a0, prompt1 # Get the multiplicand
	0x00400004	0x34240053	ori $4,$1,0x00000053	
	0x00400008	0x0c100037	jal 0x004000dc	10: jal PromptInt
	0x0040000c	0x00028821	addu $17,$0,$2	11: move $s1, $v0
	0x00400010	0x00112821	addu $5,$0,$17	13: move $a1, $s1
	0x00400014	0x24020001	addiu $2,$0,0x00000001	15: li $v0, 1
	0x00400018	0x0c10000d	jal 0x00400034	16: jal Factorial # Do multiplication
	0x0040001c	0x00029021	addu $18,$0,$2	17: move $s2, $v0
	0x00400020	0x3c011001	lui $1,0x00001001	19: la $a0, result #Print the answer
	0x00400024	0x3424006b	ori $4,$1,0x0000006b	
	0x00400028	0x00122821	addu $5,$0,$18	20: move $a1, $s2
	0x0040002c	0x0c100031	jal 0x004000c4	22: jal PrintInt
	0x00400030	0x0c100040	jal 0x00400100	24: jal Exit
	0x00400034	0x23bdfff8	addi $29,$29,0xffff...	26: addi $sp, $sp -8 # push the stack
	0x00400038	0xafa50004	sw $5,0x00000004($29)	27: sw $a1, 4($sp) #save $a1
	0x0040003c	0xafbf0000	sw $31,0x00000000($29)	28: sw $ra, 0($sp) # Save the $ra
	0x00400040	0x200a0001	addi $10,$0,0x00000001	30: addi $t2, $zero, 1
	0x00400044	0x00aa4023	subu $8,$5,$10	31: seq $t0, $a1, $t2 # if (n == 1) return

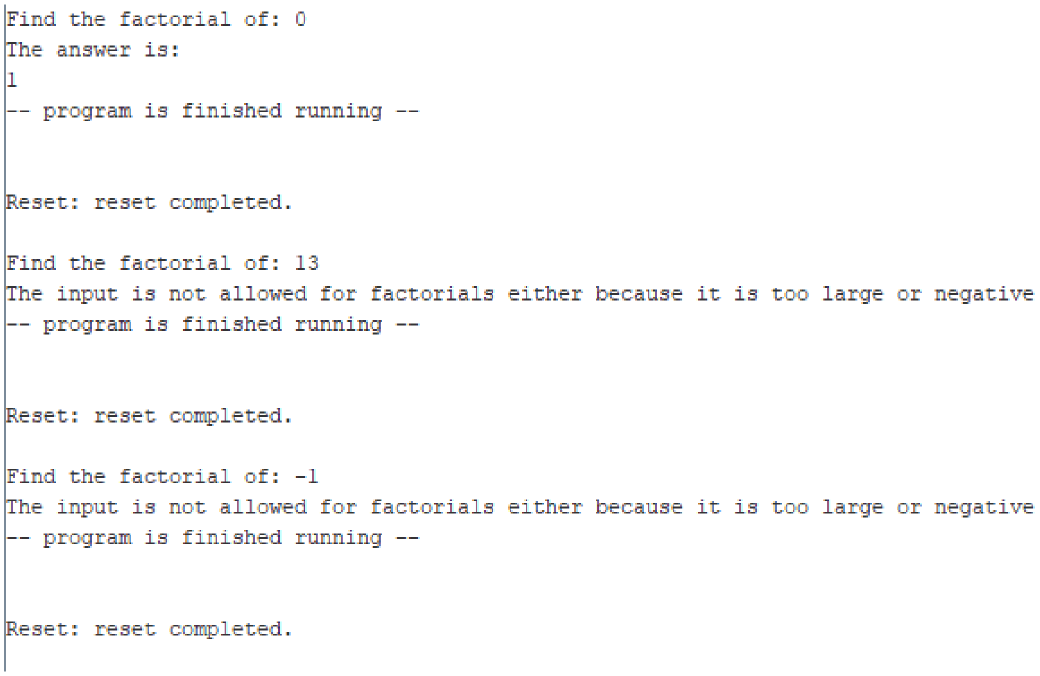
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x20656854	0x75706e69	0x73692074	0x746f6e20	0x6c6c6120	0x6465776f	0x726f6620	0x63616620
0x10010020	0x69726f74	0x20736c61	0x68746965	0x62207265	0x75616365	0x69206573	0x73692074	0x6f6f7420
0x10010040	0x72616c20	0x6f206567	0x656e2072	0x69746167	0x46006576	0x20646e69	0x20656874	0x74636166
0x10010060	0x6169726f	0x666f206c	0x5400203a	0x61206568	0x6577736e	0x73692072	0x000a203a	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

```
Find the factorial of: 0
The answer is:
1
-- program is finished running --


Reset: reset completed.

Find the factorial of: 13
The input is not allowed for factorials either because it is too large or negative
-- program is finished running --


Reset: reset completed.

Find the factorial of: -1
The input is not allowed for factorials either because it is too large or negative
-- program is finished running --


Reset: reset completed.
```

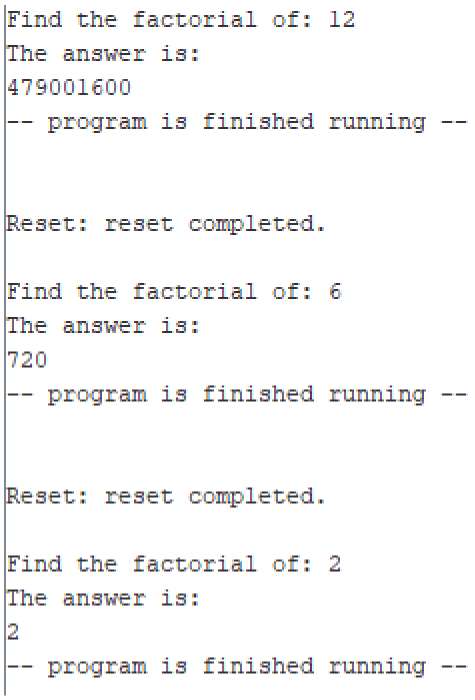
Above is a screenshot of my environment for the sorting program. I am using the MARS ide to produce this runtime.

I used sub procedure code obtained from *Introduction to MIPS Assembly Language Programming* in order to establish a working program for this problem. The edge cases for this problem were relatively straight forward. The only requirement to handle them were a series of branches. Specially ones detecting an input less than 0 and greater than 12. In the case that the initial value was 0 the program immediately returns the base case for the factorial, 1.



```
Find the factorial of: 0
The answer is:
1
-- program is finished running --


Reset: reset completed.

Find the factorial of: 13
The input is not allowed for factorials either because it is too large or negative
-- program is finished running --


Reset: reset completed.

Find the factorial of: -1
The input is not allowed for factorials either because it is too large or negative
-- program is finished running --


Reset: reset completed.
```

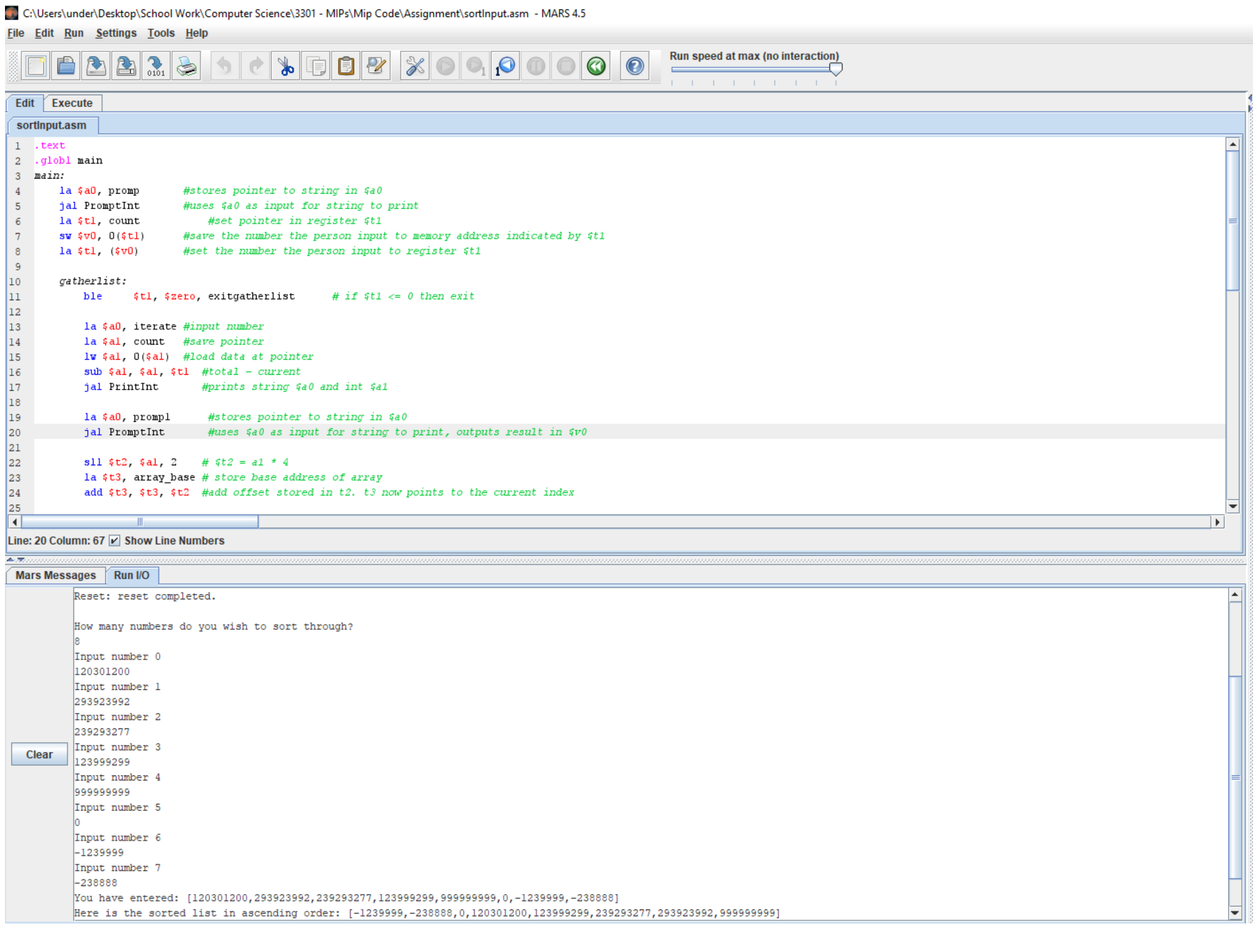
Here is a closer look at the terminal displaced on the previous page. It demonstrates the edge cases discussed earlier.



```
Find the factorial of: 12
The answer is:
479001600
-- program is finished running --


Reset: reset completed.

Find the factorial of: 6
The answer is:
720
-- program is finished running --


Reset: reset completed.

Find the factorial of: 2
The answer is:
2
-- program is finished running --
```

Finally, here are some examples of the program executing on regular values of n.

Part 2: Sorting Program

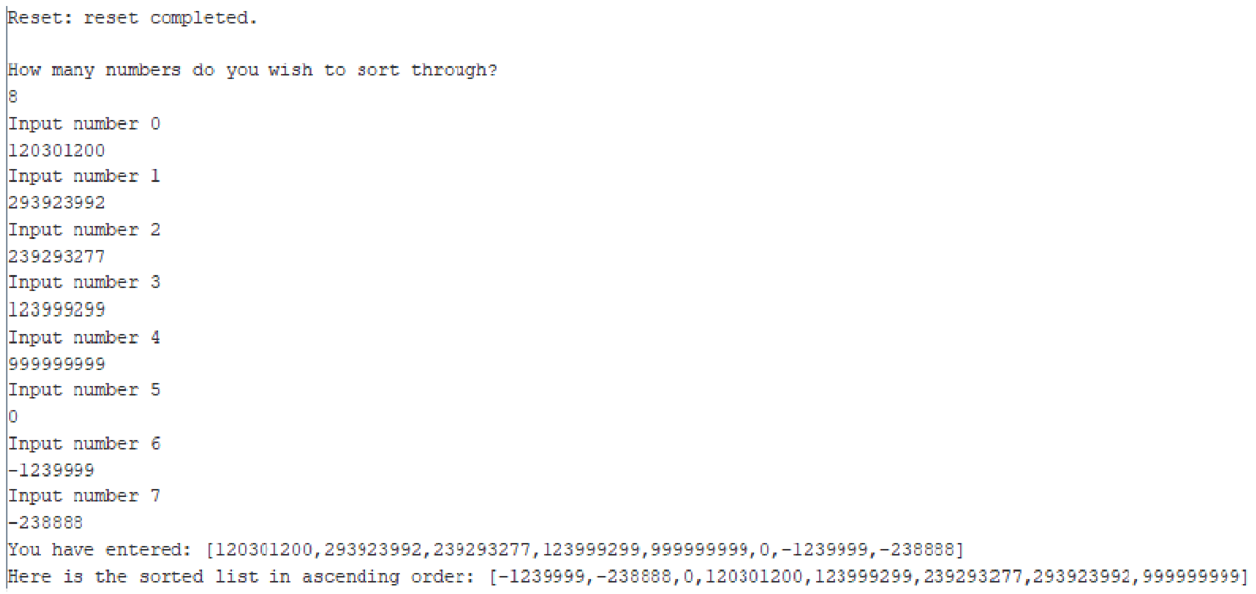

C:\Users\under\Desktop\School Work\Computer Science\3301 - MIPs\Mip Code\Assignment\sortInput.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit
Execute
sortInput.asm
Line: 20 Column: 67
Show Line Numbers
Mars Messages
Run I/O
Clear

```
.text
.globl main
main:
    la $a0, promp       #stores pointer to string in $a0
    jal PromptInt       #uses $a0 as input for string to print
    la $t1, count           #set pointer in register $t1
    sw $v0, 0($t1)      #save the number the person input to memory address indicated by $t1
    la $t1, ($v0)       #set the number the person input to register $t1

    gatherlist:
        ble     $t1, $zero, exitgatherlist      # if $t1 <= 0 then exit

        la $a0, iterate #input number
        la $a1, count   #save pointer
        lw $a1, 0($a1)  #load data at pointer
        sub $a1, $a1, $t1  #total - current
        jal PrintInt       #prints string $a0 and int $a1

        la $a0, prompl      #stores pointer to string in $a0
        jal PromptInt       #uses $a0 as input for string to print, outputs result in $v0

        sll $t2, $a1, 2    # $t2 = a1 * 4
        la $t3, array_base # store base address of array
        add $t3, $t3, $t2  #add offset stored in t2. t3 now points to the current index

```

```
Reset: reset completed.

How many numbers do you wish to sort through?
8
Input number 0
120301200
Input number 1
293923992
Input number 2
239293277
Input number 3
123999299
Input number 4
999999999
Input number 5
0
Input number 6
-1239999
Input number 7
-238888
You have entered: [120301200,293923992,239293277,123999299,999999999,0,-1239999,-238888]
Here is the sorted list in ascending order: [-1239999,-238888,0,120301200,123999299,239293277,293923992,999999999]
```

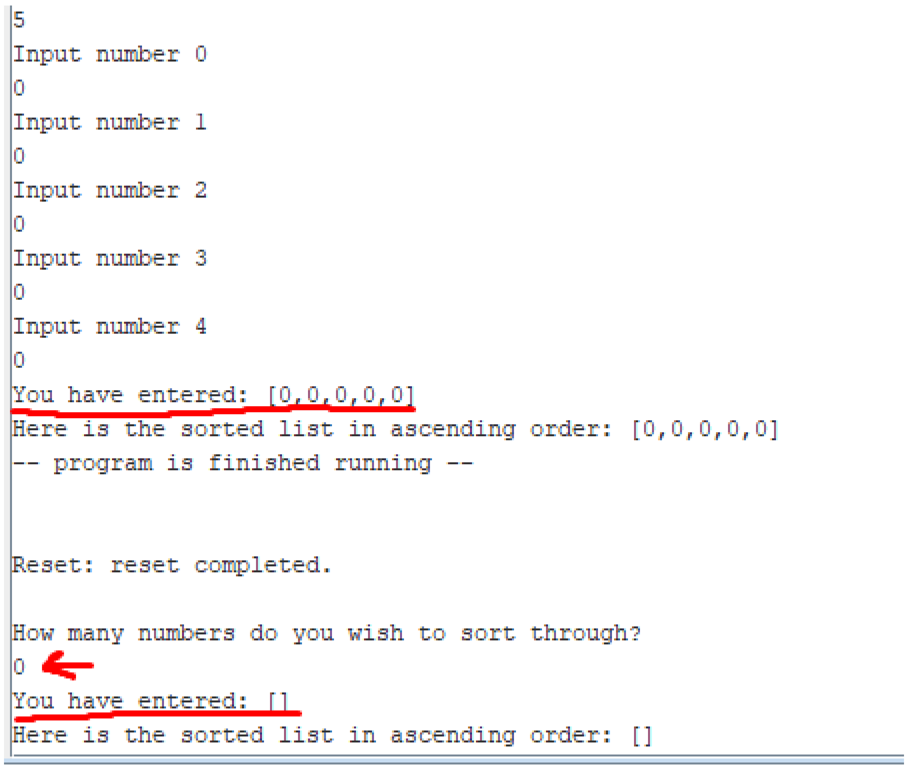
Above is a screenshot of my environment for the sorting program. I am using the MARS ide to produce this runtime.

I used sub procedure code obtained from *Introduction to MIPS Assembly Language Programming* in order to establish a working program for this problem. The difficulties I encountered were with allocating space for the array base address data to some arbitrary value, n. Therefore, I manually limited the data to 100 entries. I did not do extensive edge case testing for invalid entries. For example, if a letter is entered; the expected datatype is an integer so the program will throw an exception. This is not indicated specifically under the list of edge cases. So specifically, I will demonstrate the listed edge cases.



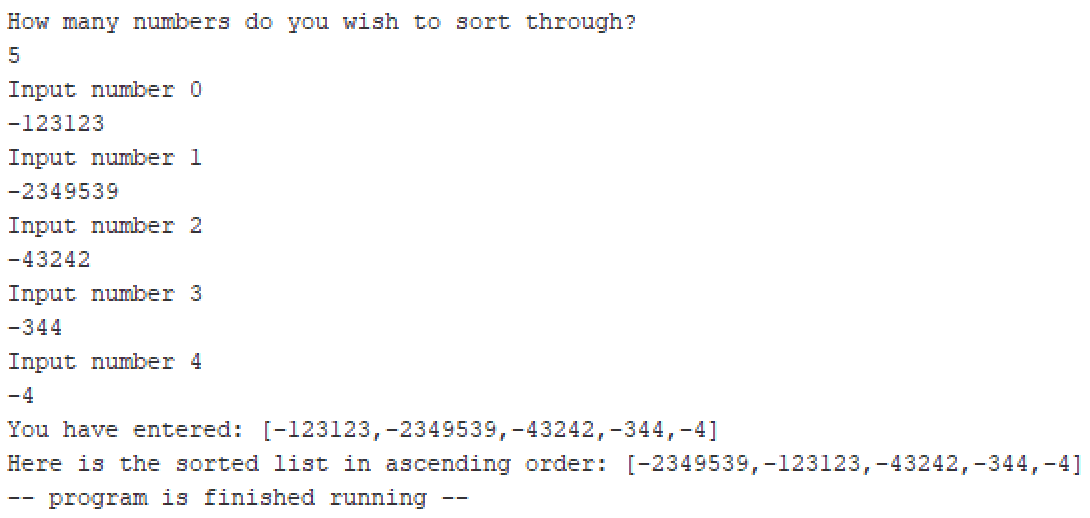
```
Reset: reset completed.

How many numbers do you wish to sort through?
8
Input number 0
120301200
Input number 1
293923992
Input number 2
239293277
Input number 3
123999299
Input number 4
999999999
Input number 5
0
Input number 6
-1239999
Input number 7
-238888
You have entered: [120301200,293923992,239293277,123999299,999999999,0,-1239999,-238888]
Here is the sorted list in ascending order: [-1239999,-238888,0,120301200,123999299,239293277,293923992,999999999]
```

Here is a closer look at the terminal, I chose to sort 8 numbers, the maximum is arbitrarily 100. This execution demonstrates large negative and positive values correctly ordered with 0 included in the dataset. If an integer is larger than the maximum held in 32 bits then the program will throw an exception.



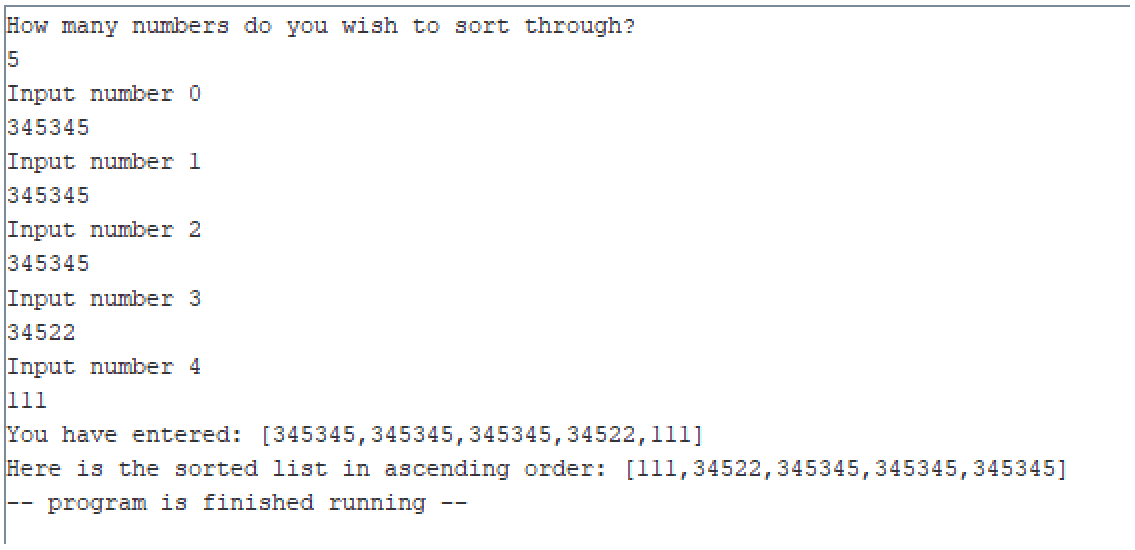
```
5
Input number 0
0
Input number 1
0
Input number 2
0
Input number 3
0
Input number 4
0
You have entered: [0,0,0,0,0]
Here is the sorted list in ascending order: [0,0,0,0,0]
-- program is finished running --


Reset: reset completed.

How many numbers do you wish to sort through?
0
You have entered: []
Here is the sorted list in ascending order: []
```

This is an example of an empty array, both in size and sum.



```
How many numbers do you wish to sort through?
5
Input number 0
-123123
Input number 1
-2349539
Input number 2
-43242
Input number 3
-344
Input number 4
-4
You have entered: [-123123,-2349539,-43242,-344,-4]
Here is the sorted list in ascending order: [-2349539,-123123,-43242,-344,-4]
-- program is finished running --
```

This is the output with only negative integers.



```
How many numbers do you wish to sort through?
5
Input number 0
345345
Input number 1
345345
Input number 2
345345
Input number 3
34522
Input number 4
111
You have entered: [345345,345345,345345,34522,111]
Here is the sorted list in ascending order: [111,34522,345345,345345,345345]
-- program is finished running --
```

And finally, only positive output.