1

```
#!/bin/sh
while [ $# -gt 1 ] #Continues executing loops until number of arguments is
greater than 1
do
     shift #Shifts all the arguments to the left until one argument left
done
echo $1 #Prints last argument
```

Test Cases:

```
obelix[18]% lastarg arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10
arg11
arg11
obelix[19]% lastarg

obelix[20]% lastarg the whole crowd goes so loud
loud
```

The cd changes the working directory to the home directory. The lastarg script is executed on all the hidden files in the working directory (the ones that start with “.”)

```
obelix[25]% cd; lastarg .*
.xsession.14-09-11
```

2

```
#!/bin/sh
echo $0 #Prints shell script file name
while [ $# -gt 1 ]; do #Repeats loop until one argument is left
	echo $1 #Prints out first argument
	shift #Shifts to the left twice to skip even-numbered arguments
	shift
done
```

```
obelix[21]% odd_prn to C or not to C that is the question
odd_prn
to
or
to
that
the
obelix[22]% odd_prn 1 2 3 4 5 6 7 8 9
odd_prn
1
3
5
7
obelix[23]% odd_prn
odd_prn
```

The cd changes the working directory to the home directory. The odd_prn script is executed on all the hidden files in the working directory (the ones that start with “.”) and outputs all the odd-numbered files.

```
obelix[24]% cd; odd_prn .*
odd_prn
.
.A*"?'\`A
.WebStorm8
.Xdefaults
.alias.rs6000
.alias.sun4m
.cache
.cshrc
.dmrc
.emacs
.gconf
.gnome
.gnome2_private
.history.sun4
.local
.macromedia
.mwmrc
.pki
.plan.txt
.recently-used
.ssh
.twmrc
```

```
.xsession
```

3

```
#!bin/sh
echo "How many columns would you like?" #Prompts user for column count
read columns #Accepts number of columns from user
j=0 #Initializes variables for nested loop
k=0
#Outputs first half of triangle
while [ $j -lt $columns ]; do #Outside loop for rows
     j=`expr $j + 1` #Increments row count
     k=0 #Resets column count to 0
     while [ $k -lt $j ]; do #Loops from 0 to column number
          echo -n "$k " #Outputs characters with spaces in between
          k=`expr $k + 1` #Increments column count
     done
     echo "" #Changes to new lines
done
#Outputs second half of triangle
while [ $j -gt $0 ]; do #Outside loop for rows, number of columns
     j=`expr $j – 1` #Decrements maximum column count
     k=0 #Resets column count to 0
     while [ $k -lt $j ]; do #Inside loop for columns
          echo -n "$k " #Outputs characters with spaces in between
          k=`expr $k + 1` #Increments column count
          done
     echo "" #Changes to new line
done
```

Note: I asked all the T.A's whether the “input during execution” portion meant using READ or accepting an argument and the first one to reply said that either was fine.

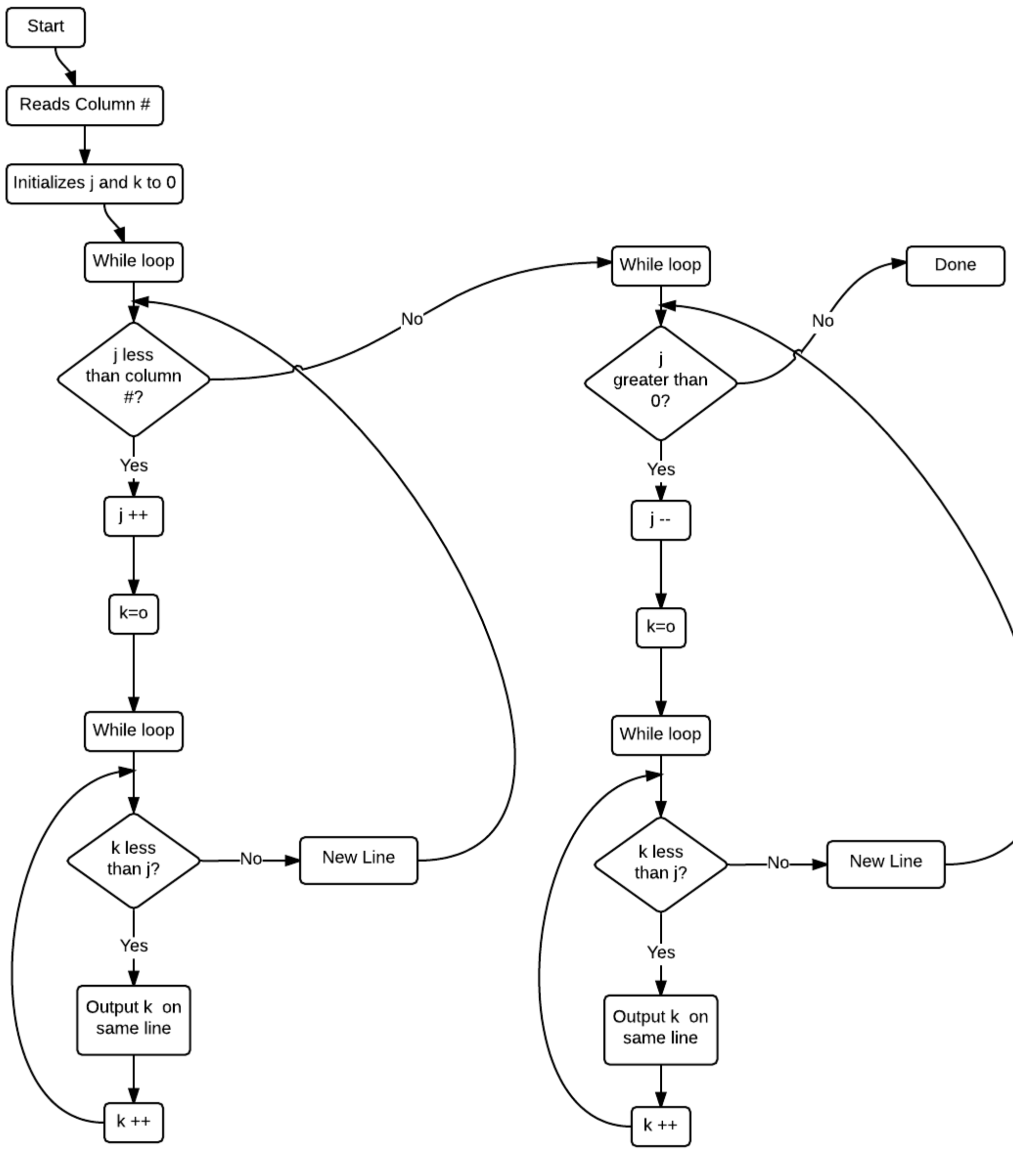
```
obelix[27]% sh pyramid
How many columns would you like?
6
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0

obelix[28]% sh pyramid
How many columns would you like?
10
0
0 1
0 1 2
0 1 2 3
```

```
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0

obelix[29]% sh pyramid
How many columns would you like?
1
0
```

Flow Chart on following page.


Start
Reads Column #
Initializes j and k to 0
While loop
j less than column #?
Yes
j ++
k=o
While loop
k less than j?
No
New Line
Yes
Output k on same line
k ++
No
While loop
j greater than 0?
No
Done
Yes
j --
k=o
While loop
k less than j?
No
New Line
Yes
Output k on same line
k ++

4

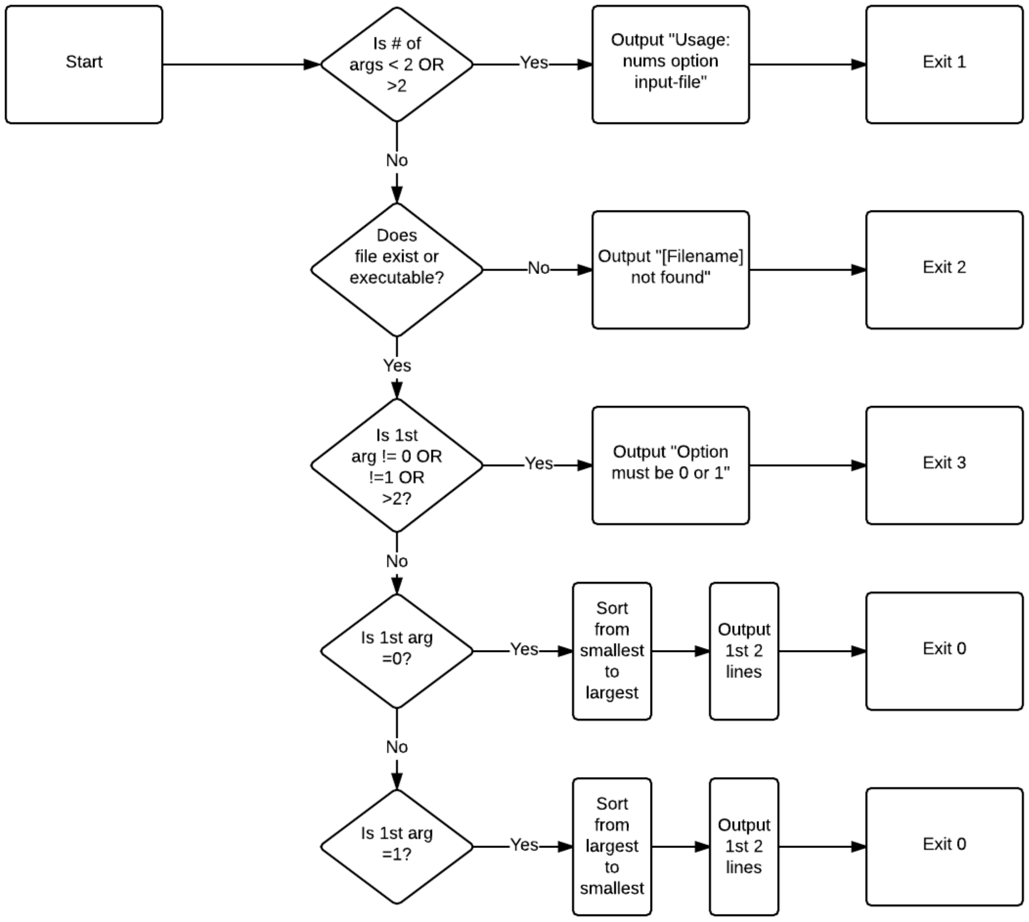
```
#!/bin/sh
if [ $# -lt 2 ]||[ $# -gt 2 ]; then #Checks if number of arguments is less
than 2 or greater than 2
     echo "Usage: nums option input-file"
     exit 1 #Exits script with exit code 1
fi
if [ -f $2 ]; then #Checks if file exists
     if [ $1 -ne 0 ]&&[ $1 -ne 1 ]&&[ $1 -gt 2 ]; then #Checks if first
argument isn't 1 or 0 or greater than 2
          echo "Option must be 0 or 1"
          exit 3
     fi
     if [ $1 -eq 0 ]; then #Checks if first argument is 0
          sort -n $2 | head -2 #Sorts and takes first 2 lines (smallest)
          exit 0
     fi
     if [ $1 -eq 1 ]; then #Checks if first argument is 1
          sort -nr $2 | head -2 #Sorts in reverse order and takes first 2
lines (largest)
          exit 0
     fi
else
     echo "$2 not found" #Outputs name of file and statement that it wasn't
found
     exit 2
fi
done
```

Test Cases:

```
obelix[40]% nums ; echo $?
Usage: nums option input-file
1
obelix[41]% nums 0 ; echo $?
Usage: nums option input-file
1
obelix[42]% nums 5 ; echo $?
Usage: nums option input-file
1
obelix[43]% nums 0 numbersfile ; echo $?
-10
-8
0
obelix[44]% nums 1 numbersfile ; echo $?
16
11
0
obelix[45]% nums numbersfile ; echo $?
Usage: nums option input-file
1
obelix[46]% nums 5 numbersfile ; echo $?
```

```
Option must be 0 or 1
3
obelix[56]% nums 0 numbersfile aaaa ; echo $?
Usage: nums option input-file
1
obelix[51]% nums 0 aaaa ; echo $?
aaaa not found
2
obelix[52]% nums 1 bbbb ; echo $?
bbbb not found
2
```

Flow Chart on Following Page:


Start
Is # of args < 2 OR >2
Yes
Output "Usage: nums option input-file"
Exit 1
No
Does file exist or executable?
No
Output "[Filename] not found"
Exit 2
Yes
Is 1st arg != 0 OR !=1 OR >2?
Yes
Output "Option must be 0 or 1"
Exit 3
No
Is 1st arg =0?
Yes
Sort from smallest to largest
Output 1st 2 lines
Exit 0
No
Is 1st arg =1?
Yes
Sort from largest to smallest
Output 1st 2 lines
Exit 0