Question 1

Code:

```
# lastarg finds and prints the last argument it is given.


#!/bin/sh


# This loop discards every argument until there is only one argument
left.


while [ $# -gt 1 ]    # Waits until there is only one argument left.

do

     shift            # Discards each argument whenever the number of
                      arguments is greater than 1.

done


if [ $# -eq 1 ]       # Verifies that there is only one argument left
                      before printing.

then

     echo "$1"        # Prints the only remaining argument.

fi
```

Cases:

No arguments:

```
obelix[121]% lastarg
```

4 arguments (covers values under 10):

```
obelix[122]% lastarg this will print success!
success!
```

13 arguments (covers values over 10):

```
obelix[123]% lastarg arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11 arg12 arg13
arg13
```

If **lastarg** is placed in your home directory, what will happen if you execute the following command?

```
cd; lastarg .*
```

Executing that command would print the name of the last hidden file if they were organized alphabetically in ascending order. This is because the input for **lastarg** are the executable file names in my home directory in alphabetical order.

Question 2

Code:

```
# odd_prn prints its command name and every odd numbered argument after it

#!/bin/sh

echo $0                    # Prints the command.

# This loop checks to make sure the number of arguments is more than zero and then prints the next argument.
# After that, it discards the most recently printed argument and the following one to successfully print each odd numbered argument.

while [ $# -gt 0 ]
do
      echo "$1"            # Prints the current argument.
      if [ $# -eq 1 ]      # Check to make sure that we don't shift
                           twice if there is only one argument left.
      then
            shift          # If the current argument is the last
                           argument left, discard it.
      else                 # Otherwise, if there is more than one
                           argument left, proceed:
            shift          # Discards the current argument.
            shift          # Discards the next (even numbered)
                           argument.
      fi
done
```

Cases:

No arguments:

```
obelix[93]% odd_prn

odd_prn
```

4 arguments (covers even numbers and proves that the code works for values under 10):

```
obelix[94]% odd_prn arg1 arg2 arg3 arg4

odd_prn

arg1

arg3
```

13 arguments (covers odd numbers and proves that the code works for values greater than 10):

```
obelix[94]% odd_prn arg1 arg2 arg3 arg4

odd_prn

arg1

arg3
```

If **odd_prn** is placed in your home directory, what will happen if you execute the following command?

```
cd; odd_prn .*
```

Executing that command would print the name of the command, followed by that of the first and every second hidden file after that in a new line each time, if they were organized alphabetically in ascending order. This is because the input for **odd_prn** are the executable file names in my home directory in alphabetical order.

Question 3

Code:

```
# num_pyramid prints a horizontal pyramid of specified width


#!/bin/sh


read input                       # Assigns an input to variable "input.

width=0                          # Initializes width with a value of 0.
                                 "Width" defines the number of columns
                                 that each row will have printed.


while [ $width -lt $input ]      # This loop creates the first half of
                                 the pyramid (ascending), up until
                                 “input”-1.

do

column=0

     while [ $column -le $width ]     # This loop creates each row
                                      individually by incrementing
                                      "column" to a maximum specified
                                      by "width."

     do

          echo -n "$column "

          column=`expr $column + 1`   # Increments "column" by one
                                      each time.

     done


     echo                        # Prints an empty line to begin the
                                 next row.

     width=`expr $width + 1      # Increments "width" to accommodate
                                 for a bigger row.
```

```
done


width=`expr $width – 2            # Decrements "width" by 2 to prepare
                                  it for the bottom half of the pyramid.

                                  # It does so because, at this point,
                                  "width" is equal to n and it needs to
                                  be at n - 2 for the next row.

while [ $width -ge 0 ]            # This loop creates the bottom portion
                                  of the pyramid by decrementing "width"
                                  back down to 0.

do

column=0

      while [ $column -le $width ]     # Like the above loop, this loop
                                       creates each row individually by
                                       incrementing "column to a
                                       maximum of "width".

      do

            echo -n "$column "

            column=`expr $column + 1`  # Increments "column" by one
                                       each time.

      done


      echo                        # Prints an empty line to begin the
                                  next row.

      width=`expr $width - 1`     # Decrements "width" to prepare it for
                                  the next row.

done
```

Cases:

argument is 0 (zero columns):

```
obelix[58]% num_pyramid
0
```

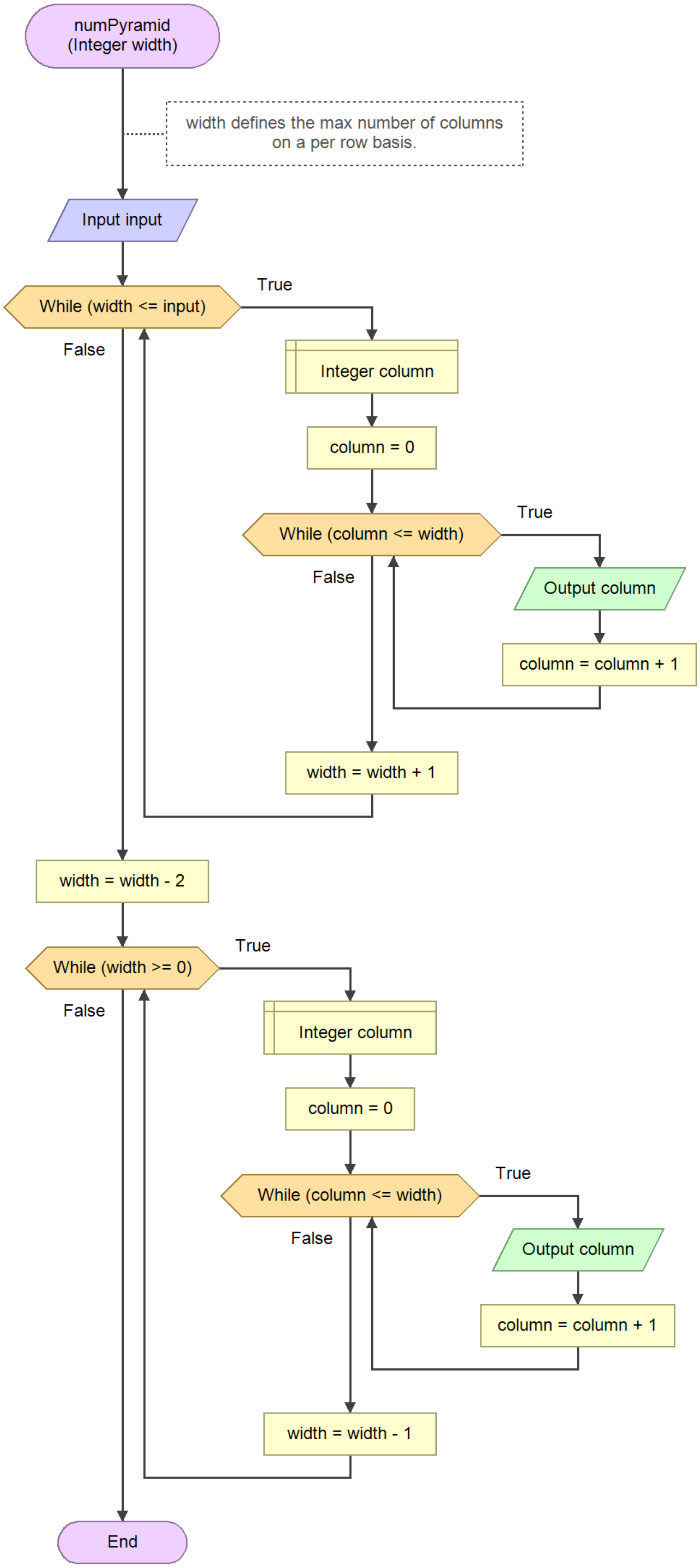
argument is 4 (greater than zero):

```
obelix[59]% num_pyramid
4
0
0 1
0 1 2
0 1 2 3
0 1 2
0 1
0
```

argument is 6 (example given):

```
obelix[60]% num_pyramid
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
```

Flowchart:


numPyramid
(Integer width)
width defines the max number of columns
on a per row basis.
Input input
While (width <= input)
True
False
Integer column
column = 0
While (column <= width)
True
False
Output column
column = column + 1
width = width + 1
width = width - 2
While (width >= 0)
True
False
Integer column
column = 0
While (column <= width)
True
False
Output column
column = column + 1
width = width - 1
End

Question 3

Code:

```
# nums 0 finds the first and second smallest numbers inside of a file.

# nums 1 finds the first and second largest numbers inside of a file.


#!/bin/sh


if [ $# -ne 2 ]                  # Checks to make sure that there are
                                 two arguments present.

then

     echo "Usage: nums option input-file"      # If not, displays a
                                               warning message.

     exit 001                    # And exit with status 001
                                 (unsuccessful).

fi


if [ ! -f  "$2" ]                # Checks to make sure that the input
                                 file exists.

then

     echo "input-file not found”      # If not, display a warning
                                      message.

     exit 002                    # And exit with status 002
                                 (unsuccessful).

fi


if [ $1 -eq 0 ]                  # Checks to see if the first argument
                                 is "0".

then

     sort -nu $2 | head -2           # If yes, sort the inputted file
                                     (argument 2) numerically in
                                     ascending order and display the
                                     first two lines.
```

```
elif [ $1 -eq 1 ]                # Otherwise, checks to see if the
                                 first argument is "1".

then

     sort -nu $2 | tail -2            # If yes, sort the inputted file
                                      (argument 2) numerically in
                                      ascending order and display the
                                      last two lines.

else                             # Finally, if the number is neither 0
                                 nor 1,

     echo "Option must be 0 or 1."         # Display a warning
                                           message.

     exit 003                    # And exit with status 003

fi
```

Cases:

nums ; echo $?

```
obelix[13]% nums ; echo $?
Usage: nums option input-file
1
```

nums 0; echo $?

```
obelix[13]% nums ; echo $?
Usage: nums option input-file
1
```

nums 5; echo $?

```
obelix[19]% nums 5 ; echo $?
Usage: nums option input-file
1
```

nums 0 numbersfile; echo $?

```
obelix[20]% nums 0 numbersfile; echo $?
-10
-8
0
```

nums 1 numbersfile; echo $?

```
obelix[21]% nums 1 numbersfile; echo $?
11
16
0
```

nums numbersfile; echo $?

```
obelix[23]% nums numbersfile; echo $?

Usage: nums option input-file

1
```

nums 5 numbersfile; echo $?

```
obelix[24]% nums 5 numbersfile; echo $?

Option must be 0 or 1.

3
```

nums 0 numbersfile aaaa; echo $?

```
obelix[25]% nums 0 numbersfile aaaa; echo $?

Usage: nums option input-file

1
```

nums 0 aaaa; echo $?

```
obelix[26]% nums 0 aaaa; echo $?

input-file not found

2
```

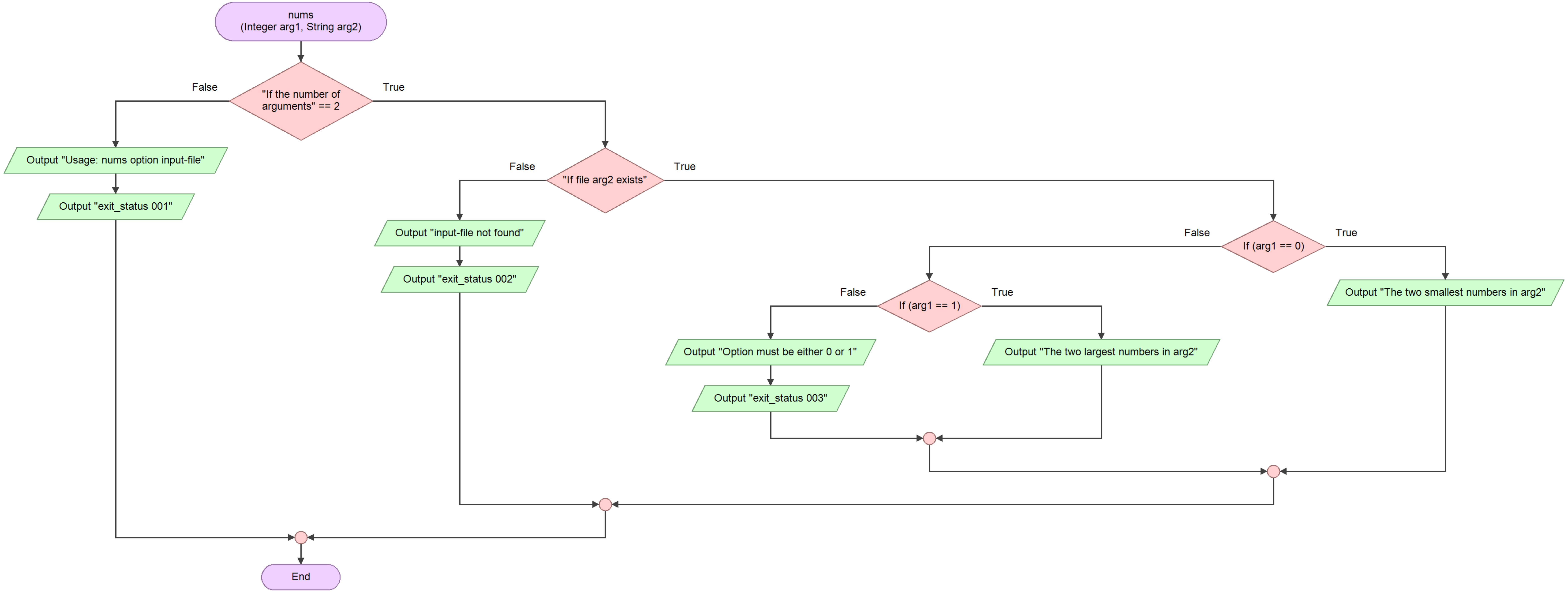
nums 1 bbbb; echo $?

```
obelix[27]% nums 1 bbbb; echo $?

input-file not found

2
```

Flowchart:


nums
(Integer arg1, String arg2)
"If the number of arguments" == 2
False
True
Output "Usage: nums option input-file"
Output "exit_status 001"
"If file arg2 exists"
False
True
Output "input-file not found"
Output "exit_status 002"
If (arg1 == 0)
False
True
Output "The two smallest numbers in arg2"
If (arg1 == 1)
False
True
Output "Option must be either 0 or 1"
Output "exit_status 003"
Output "The two largest numbers in arg2"
End