Q1

Code

```
#!/bin/sh

#Checks the value of $#. If it is zero the program stops and doesn't
return anything.
#If the value is more than zero, it proceeds to excute the else
statement.

if [ $# -lt 1 ]
then
      :
else
      #Checks if the value of $# is more than one.
      #Shifts the of #$ to the next string.

      while [ $# -gt 1 ]
      do
           shift
      done

      #Echos the value of the last string $1
      echo $1
fi
```

```
cd; lastarg .*
```
.xsession

Case 1:
```
cd ; lastarg* 1 2 3 4 5 6 7 8 9 10 11
```
11

Case 2:
```
cd ; lastarg* arg1 arg2 ag3
```
ag3

Case 3:
```
cd ; lastarg* a b c d e f g
```
g

$$

Q2

Code

```
#!/bin/sh

#Echos the name of the program.
echo $0
```

```
#Checks if the value of $# is more than zero. If it is the program
terminates.
#If the value is more than zero, it proceeds to excute the else
statement.
if [ $# -lt 1 ]
then
      :
else

      #Echos the first word.
      echo $1

      #Runs while the value of $# is more than two.
      while [ $# -gt 2 ]
      do
            #Shifts two strings to the right.
            shift
            shift
            #Echos the odd strings.
            echo $1
      done
fi
```

```
Cd; odd_prn .*
```

```
Odd_prn
.
.A*"?'\`A
.Xauthority
._Library
.alias.sun4
.alias.sun4u
.cups
.forward
.login
.plan
.solregis
.twmrc
.xsession
```

Trail 1:

```
odd_prn to C or not to C that is the question
```

```
odd_prn
to
or
to
that
the
```

Trail 2:

```
odd_prn give x me x a x 100
```

```
odd_prn
give
me
a
100
```

$$

Q3

Code

```
#!/bin/sh

#Sets the variables i, x, y, z to to the input, zero, zero, and value
of i plus one respectively
i=$1
x=0
y=0
z=`expr $i + 1`

#Nested while loop that does the first half of the triangle.
while [ $z -gt $x ]
do
      while [ $x -gt $y ]
      do
            #Prints the values of y on the same line
            echo -n $y
            echo -n " "
            y=`expr $y + 1`
      done
      #Resets the value of y, gives a new value to x, and starts a new
line.
      y=0
      x=`expr $x + 1`
      echo
done

#Gives new values to x and z.
x=`expr $x - 2`
z=`expr $i - 1`

#Nested while loop that does the second half of the triangle.
while [ $x -gt 0 ]
do
      y=0
      while  [ $y -lt $z ]
      do
            #Prints the values of y on the same line
            echo -n $y
            echo -n " "
            y=`expr $y + 1`
      done
      #Gives new values to x and z, and starts a new line.
      x=`expr $x - 1`
      z=$x
      echo
done
```

Pseudo Code

```
i=entry
x=0
y=0
z= i+1

while [ z > x ]
do
      while [ x > y ]
      do
            print y
            y=y+1
      done
      y=0
      x=x+1
      print a new line
done

x=x-2
z=i-1
while [ x > 0 ]
do
      y=0
      while  [ y < z ]
      do
            print y
            y=y+1
      done
      x=x-1
      z=x
      print a new line
done
```

```
Input:
trinum 9

Output:
```

```
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
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
```

$$

Q4

Code

```
#!/bin/sh

#Checks if the number of arguments is 2, if not it displays an error
message.
if [ $# -ne 2 ]
then
     echo "Usage: nums option input-file"
#Checks if the value of argument 2 (the file) is in the directory, if
not it displays an error message.
elif [ ! -f $2 ]
then
     echo "input-file not found"
#Checks if the value of the first argument is 0 or 1, if not it
displays an error message.
elif [ $1 -ne 0 -a $1 -ne 1 ]
then
     echo "Option must be 0 or 1"
else
     #Sets the value of file to the second argument.
     file=$2
     #Reads the file into the list 'lst'
     lst=`cat $file`
     #Sets the values of num1 and num2 to zero
     num1=0
     num2=0

     #Checks if the ueser wants the smallest or the largest numbers
     if [ $1 -eq 0 ]
     then
          #A for loop that looks through the lst for the smallest
values
          for i in $lst
          do
```

```
                #Checks if the value of i is less than the value of
num2
                if [ $i -lt $num2 ]
                then
                     #Checks if the value of i is less than the value
of num1
                     if [ $i -lt $num1 ]
                     then
                          num2=$num1
                          num1=$i
                     #Checks if the value of i is less than the value
of num2
                     elif [ $i -lt $num2 ]
                     then
                          num2=$i
                     fi
                fi
           done
           #Prints the results
           echo "The 2nd smallest in the list number is $num2"
           echo "The smallest number in the list is $num1"
     else
           #A for loop that looks through the lst for the largest
values
           for i in $lst
           do
                #Checks if the value of i is larger than the value of
num2
                if [ $i -gt $num2 ]
                then
                     #Checks if the value of i is larger than the
value of num1
                     if [ $i -gt $num1 ]
                     then
                          num2=$num1
                          num1=$i
                     #Checks if the value of i is larger than the
value of num2
                     elif [ $i -gt $num2 ]
                     then
                          num2=$i
                     fi
                fi
           done
           #Prints the results
           echo "The 2nd largest in the list number is $num2"
           echo "The largest number in the list is $num1"
     fi
fi
```

Pseudo Code

```
if [ Number of arguments not equal to 2]
```

```
then
    print "Usage: nums option input-file"
else if [ does not find the file provided by the second argument in
the directory ]
then
    print "input-file not found"
else if [ the first argument is not equal to 0 or 1 ]
then
    print "Option must be 0 or 1"
else
    file=second argument value
    lst= read file
    num1=0
    num2=0

    if [ first argument equal to 0 ]
    then
        for i in list length
        do
            if [ list value at line i less than num2]
            then
                if [list value at line i less than num1]
                then
                    num2=value of num1
                    num1=value of the list at line i
                else if [list value at line i less than num2]
                then
                    num2= value of the list at line i
                fi
            fi
        done
        print "The 2nd smallest in the list number is num2s value"
        print "The smallest number in the list is num1s value"
    else
        for i in list length
        do
            if [list value at line i more than num2]
            then
                if [list value at line i more than num1]
                then
                    num2=value of num1
                    num1=value of the list at line i
                else if [list value at line i more than num2]
                then
                    num2= value of the list at line i
                fi
            fi
        done
        print "The 2nd largest in the list number is num2s value"
        print "The largest number in the list is num1s value"
    fi
fi
```

```
Input: nums ; echo $?

Output:
```

```
Usage: nums option input-file
0
```

```
Input: nums 0; echo $?

Output:
```

```
Usage: nums option input-file
0
```

```
Input: nums 5; echo $?

Output:
```

```
Usage: nums option input-file
0
```

```
Input: nums 0 numbersfile; echo $?

Output:
```

```
The 2nd smallest in the list number is -8
The smallest number in the list is -10
0
```

```
Input: nums 1 numbersfile; echo $?

Output:
```

```
The 2nd largest in the list number is 11
The largest number in the list is 16
0
```

```
Input: nums numbersfile; echo $?

Output:
```

```
Usage: nums option input-file
0
```

```
Input: nums 5 numbersfile; echo $?

Output:
```

```
Option must be 0 or 1
0
```

```
Input: nums 0 numbersfile aaaa; echo $?
```

Output:

```
Usage: nums option input-file
0
```

```
Input: nums 0 aaaa; echo $?
```

Output:

```
input-file not found
0
```

```
Input: nums 1 bbbb; echo $?
```

Output:

```
input-file not found
0
```