Question #1

Write a Bourne shell script lastarg, which takes 0 or more arguments and prints the last (rightmost) argument in the argument list. You can assume that arguments will be made up of letters and digits only.

Program:

```
#!/bin/sh       #initializing shell script
args=${!#}      #defining args as last entered argument
echo $args      #printing args.
```

If lastarg is placed in your home directory, what will happen if you execute the following command? Explain why you got this output.

- **cd; lastarg .***

  Input:
  obelix[22]% cd;
  obelix[23]% sh lastarg .*

  Output:
  .xsession.14-09-11

- When using the following commands lastarg will print out a hidden file in the home directory. This happens because the program will process all the hidden files and print out the last entered.

Test Cases	Input	Output
#1	obelix[11]% sh lastarg	lastarg
#2	obelix[13]% sh lastarg arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11	arg11
#3	obelix[16]% sh lastarg 01230ofowei 0390 ijww9 9i09i3dii	9i09i3dii

Question #2

Write a Bourne shell script odd_prn, which echoes its shell script file name as well as the values of its odd arguments. Even arguments should be ignored. Each value should be echoed in a separate line. You can assume that arguments will be made up of letters and digits only.

Program:

```
#!/bin/sh
X=$0
echo $X
#setting x to position zero so it will output program if no arguments follow
#printing $X which will be all arguments that meet condition

while [ $# -gt 0 ]; do
X=$1
echo $X
#while loop sets condition that position must be greater then zero
#set X to position one to print
#print position X which will be the first position
#two shifts will replace position $1 with Position $3 and continue to print position $1 until no arguments are left.
#this will print all odd arguments because of the double shift.

shift
shift
done
```

Test Case	Input	Output
#1	obelix[17]% bash odd_prn	odd_prn
#2	obelix[18]% bash odd_prn 1 2 3 4 5 6 7 8 9	odd_prn 1 3 5 7 9
#3	obelix[19]% bash odd_prn sdasd 1212e dq 32d 23d23d2 322d	odd_prn sdasd dq 2d 322d

#4	obelix[20]% bash odd_prn arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11	odd_prn arg1 arg3 arg5 arg7 arg9 arg11

If odd_prn is placed in your home directory, what will happen if you execute the following command? Explain why you got this output.

- **cd; odd_prn .***

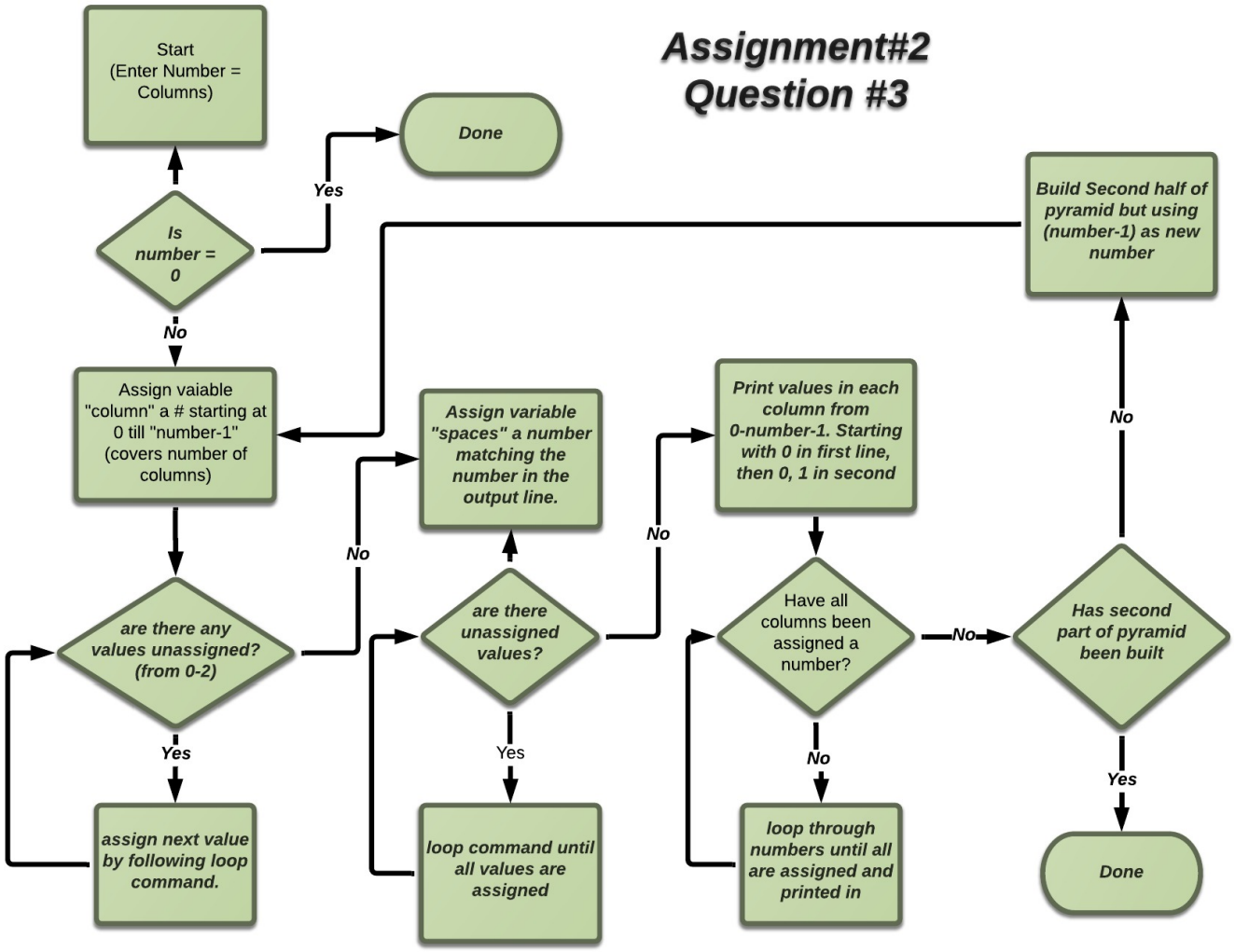
  Input:
  obelix[12]% cd;
  obelix[13]% bash odd_prn .*
  Output:
  odd_prn
  .
  .ICEauthority
  .Xauthority
  .alias.rs6000
  .alias.sun4m
  .cache
  .cshrc
  .dmrc
  .emacs
  .forward
  .gconfd
  .gnome2
  .gnupg
  .history.x86_64-linux
  .login
  .mwmrc
  .plan
  .recently-used
  .ssh
  .thunderbird
  .viminfo
  .xsession
  .xsession.14-09-11

- By using the commands above, assuming odd_prn is in the home directory, it will collect all hidden files in the directory and then output every item in position $1. After running through the program and shifting each item, to continue to output every odd positioned item, until no files are left.

Question 3

Draw a flow chart and write a Bourne shell script that causes the following output (below) to be displayed. Note that, there is a single space between each value. The number of column should be taken as an input during execution.

Flow Chart:



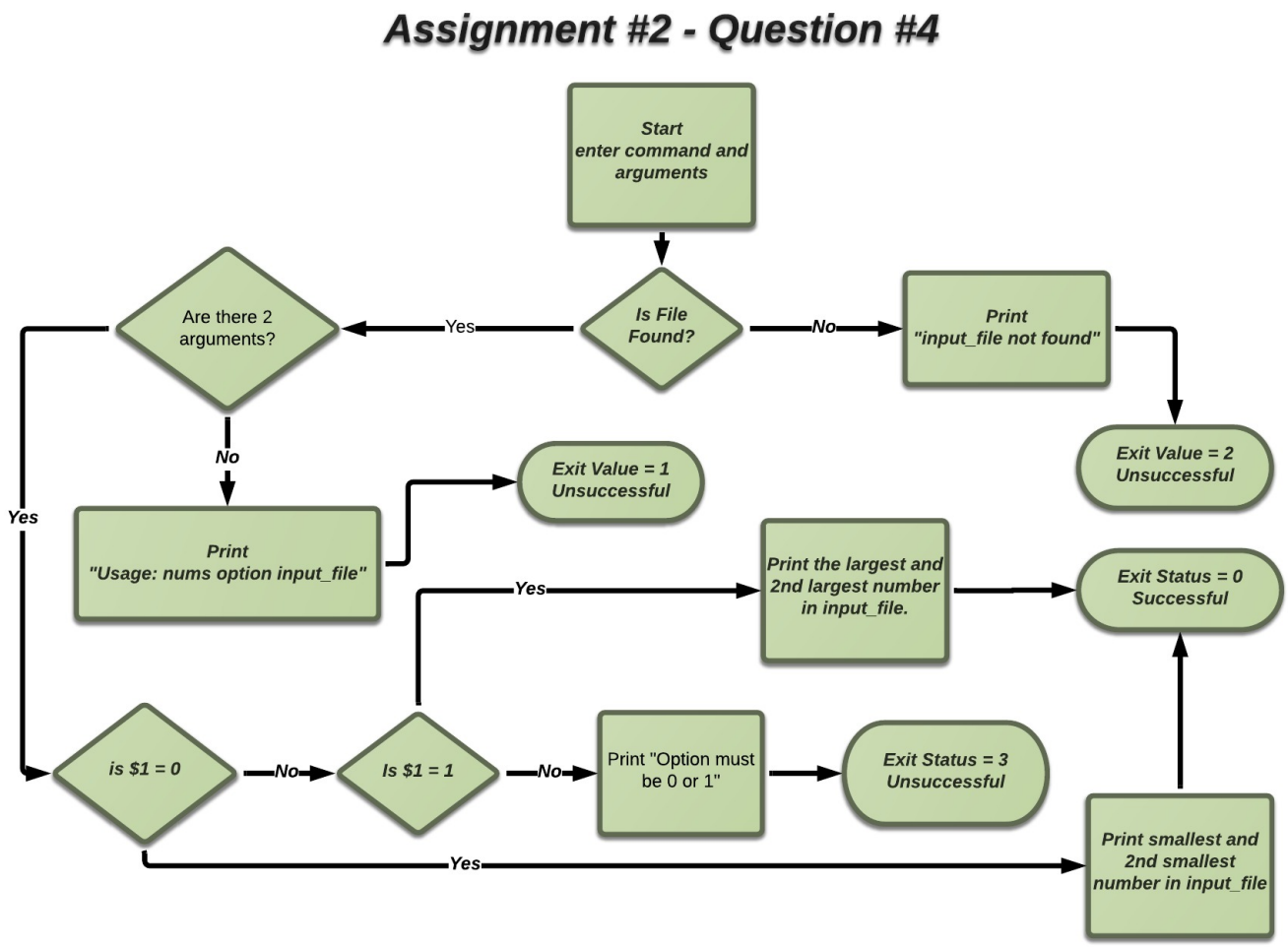
Program:

```
#!/bin/bash
#Taking input the number is how many columns will be created
echo "Enter Number:"
read number
#Outer loop for printing number of column in first half of pyramid
for((column=0; column<=number-1; column++))
do
#Loop for printing required spaces of first half of pyramid
for((spaces=0;spaces<=number; spaces++))
do
echo -ne " "
done
#Loop for printing 1st part of pyramid
for((j=0;j<=column; j++))
do
echo -ne "$j "
done
echo
done
#Outer loop for printing number of rows 2nd part in pyramid
for((column2=column-1; column2>=0; column2--))
do
#Loop for printing required spaces second half of pyramid
for((spaces2=number;spaces2>=0; spaces2--))
do
echo -ne " "
done
#Loop for printing 2nd part of pyramid
for((i=0;i<=(column2-1); i++))
do
echo -ne "$i "
done
#echo for printing new line
echo
done
```

Test Cases	Input	Output
#1	obelix[12]% bash pyramid Enter Number: 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
#2	obelix[14]% bash pyramid Enter Number: 3	0 0 1 0 1 2 0 1 0
#4	Obelix[15]% bash pyramid Enter Number: 0	(no output)

Question #4

Flow Chart



Program

```
#!/bin/bash
#checks to see that there are no more then 2 arguments
if [ "$#" -gt 2 ];then
      echo "Usage: nums option input_file"
      exit 1
fi
#checks to see that there are no less then 2 arguments
if [ "$#" -lt 2 ];then
      echo "Usage: nums option input_file"
      exit 1
fi

#checks to see that the file exists
if [ ! -s "$2" ];then
      echo "input_file not found."
      exit 2
fi

#checks to see that the option entered is 0 or 1
if [ "$1" -gt 1 ];then
      echo "Option must be 0 or 1"
      exit 3

fi
#if parameters are all correct and option is 0 then sort file smallest
to greatest and print first two lines
if [ "$1" -eq 0 ];then
      sort -nk1 $2 | head -2
      exit 0
fi

##if parameters are all correct and option is 1 then sort file
greatest to smallest and print first two lines
if [ "$1" -eq 1 ];then
      sort -rnk1 $2 | head -2
      exit 0
fi
```

Test Case	Input	Output
#1	obelix[15]% bash nums obelix[16]% echo $?	Usage: nums option input_file 1
#2	obelix[17]% bash nums 0 obelix[18]% echo $?	Usage: nums option input_file 1
#3	obelix[19]% bash nums 5 obelix[20]% echo $?	Usage: nums option input_file 1
#4	obelix[21]% bash nums 0 numbersfile obelix[22]% echo $?	-10 -8 0
#5	obelix[23]% bash nums 1 numbersfile obelix[24]% echo $?	16 11 0
#6	obelix[25]% bash nums numbersfile obelix[26]% echo $?	Usage: nums option input_file 1
#7	obelix[27]% bash nums 5 numbersfile obelix[28]% echo $?	Option must be 0 or 1 3
#8	obelix[29]% bash nums 0 numbersfile aaaa obelix[30]% echo $?	Usage: nums option input_file 1
#10	obelix[31]% bash nums 0 aaaa obelix[32]% echo $?	input_file not found. 2
#11	obelix[33]% bash nums 0 bbbb obelix[34]% echo $?	input_file not found. 2