Q1

```
#!/bin/sh
# loop while the number of args is > 1
while [ $# -gt 1 ]
do
# shift the arguments to the left
 shift
done
# return the last argument left over
echo $1
```

Two test cases I used to demonstrate possible options in my program are the case where there are 10+ parameters and the case where there are no parameters. With no parameters, the shell script returned no output and for 10+ parameters the script returned the last argument, which is correct.

The command **cd; lastarg .*** returns the last hidden file in the home directory. This is because it has the parameter for all files starting with a ".".

Q2

```
#!/bin/sh
# echo the shell script name
echo $0
# loop while the number of args is > 1
while [ $# -gt 1 ]
do
# echo the current first argument
 echo $1
# shift the arguments to the left twice
 shift
 shift
done
# echo the last arg left in position 1
echo $1
```

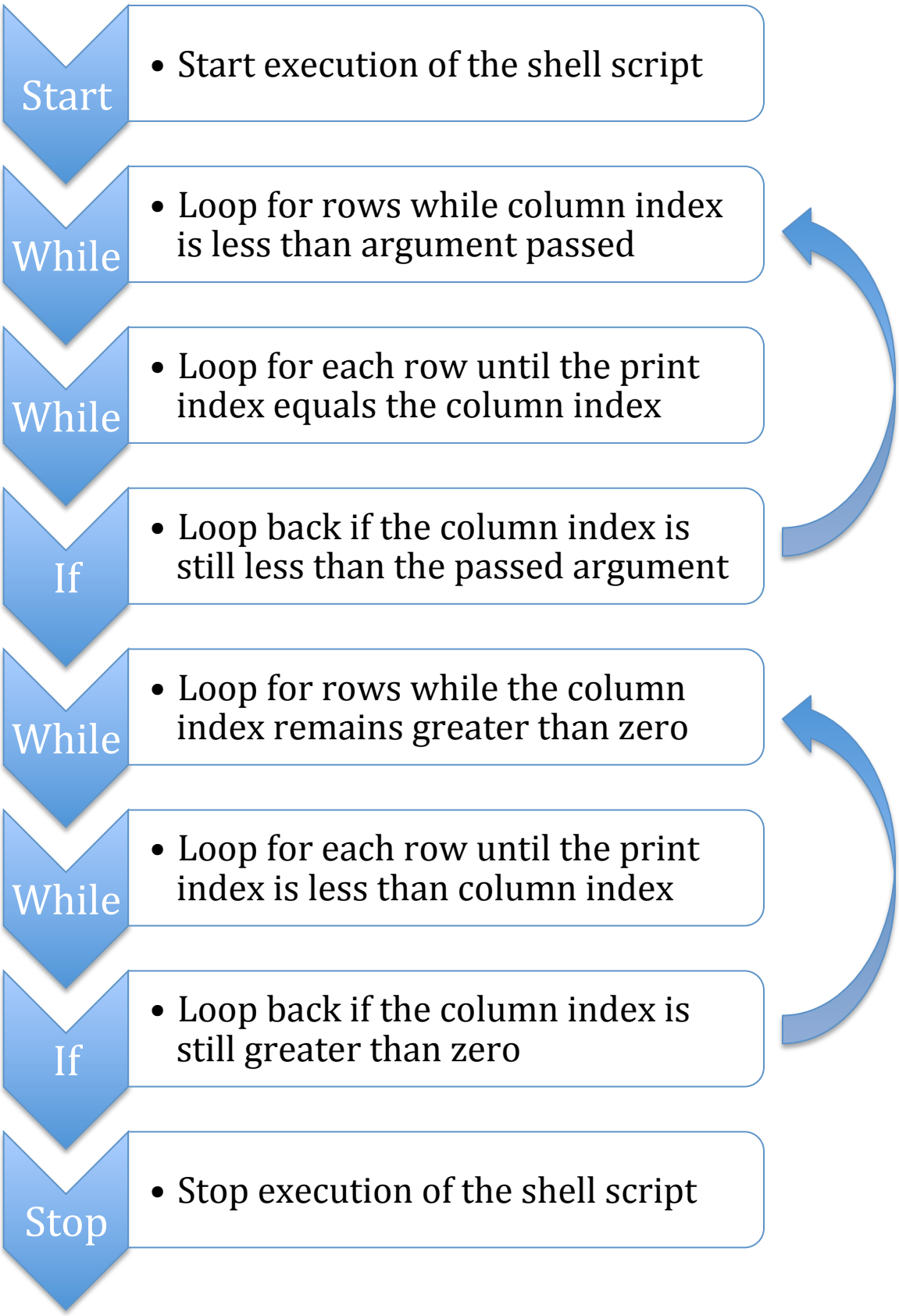
Two test cases I used to demonstrate possible options in my program are the case where there are 10+ parameters and the case where there are no parameters. With no parameters, the shell script returned only the shell script name and for 10+ parameters the script returned all of the odd arguments on separate lines, which is correct.

The command **cd; odd_prn .*** returns all of the oddly positioned hidden files in the home directory. This is because it has the parameter for all files starting with a ".".

Q3

```
#!/bin/sh
# declare my two index variables
colIndex=0
printIndex=0
# loop until there are the same number of columns as the argument
while [ $colIndex -lt $1 ]
do
 # loop until the print index equals the column index
 while [ $printIndex -le $colIndex ]
 do
  # output the current print index number and a space
  echo -n $printIndex ' '
  # increment the print index for the next term in the row
  printIndex=`expr $printIndex + 1`
 done
 # output a blank line to prepare for the next line
 echo
 # reset print index to 0
 printIndex=0
 # increment the column index for the next row
 colIndex=`expr $colIndex + 1`
done
# decrease colIndex by one to start the smaller print lines
colIndex=`expr $colIndex - 1`
# loop until the columns index is down to zero
while [ $colIndex -gt 0 ]
do
 # loop until the print index is one below the max column index
 while [ $printIndex -lt $colIndex ]
 do
  # output the current print index number and a space
  echo -n $printIndex ' '
  # increment the print index for the next term in the row
  printIndex=`expr $printIndex + 1`
 done
 # output a blank line to prepare for the next line
 echo
 # reset print index to 0
 printIndex=0
 # increment the column index down for the next row
 colIndex=`expr $colIndex - 1`
done
```

Two test cases I used to demonstrate possible options in my program are the case where I used a high integer like 10 as the argument and I also tested with 0. In both cases the script returned that integer value of columns, which is correct.


Start
• Start execution of the shell script
While
• Loop for rows while column index is less than argument passed
While
• Loop for each row until the print index equals the column index
If
• Loop back if the column index is still less than the passed argument
While
• Loop for rows while the column index remains greater than zero
While
• Loop for each row until the print index is less than column index
If
• Loop back if the column index is still greater than zero
Stop
• Stop execution of the shell script

Q4

```
#!/bin/sh
# check to make sure there are only 2 arguments
if [ $# -ne 2 ]; then
 echo "Usage: nums option input-file"
 exit 1 # exit code
else
 # check to make sure the second argument is a file and it exists
 if [ ! -f "$2" ]; then
  echo "input-file not found"
  exit 2
 else
  # check to make sure the first arguments is a 0 or 1
  if [ $1 -ne 0 ] && [ $1 -ne 1 ]; then
   echo "Option must be 0 or 1"
   exit 3
  else
   # output two smallest numbers from file
   if [ $1 -eq 0 ]; then
    sort -n $2 | head -2
   fi
   # output two largest numbers from file
   if [ $1 -eq 1 ]; then
    sort -n $2 | tail -2
   fi
   exit 0
  fi
 fi
fi
```

Test Cases:

Command	Output
nums; echo $?	Usage: nums option input-file 1
nums 0; echo $?	Usage: nums option input-file 1
nums 5; echo $?	Usage: nums option input-file 1
nums 0 numbersfile; echo $?	-10 -8 0
nums 1 numbersfile; echo $?	11 16 0
nums numbersfile; echo $?	Usage: nums option input-file 1
nums 5 numbersfile; echo $?	Option must be 0 or 1 3
nums 0 numbersfile aaaa; echo $?	Usage: nums option input-file

	1
nums 0 aaaa; echo $?	input-file not found 2
nums 1 bbbb; echo $?	input-file not found 2