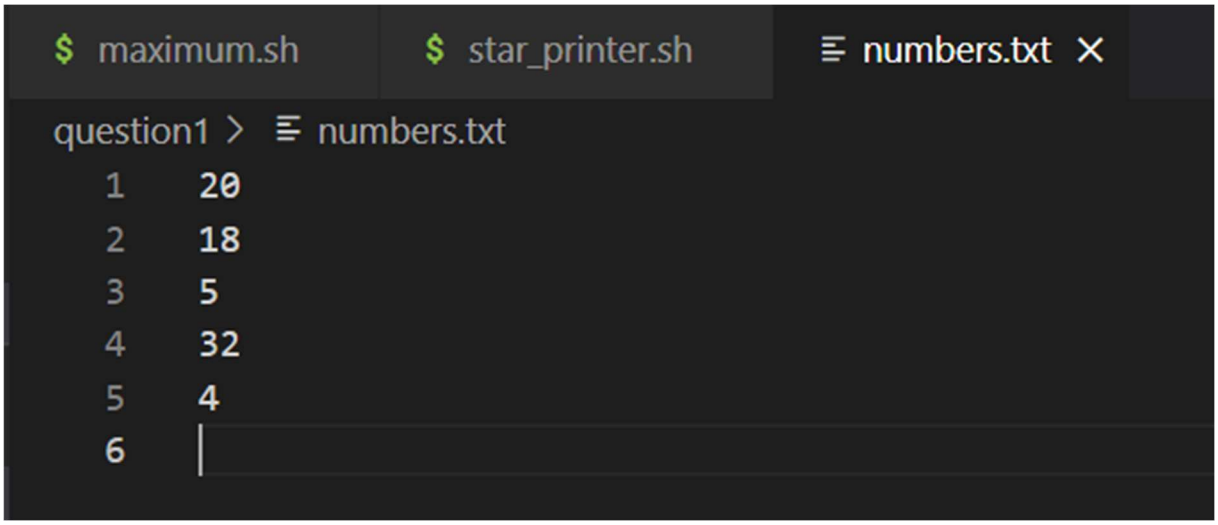
Question 1)

First, I have written the program to take an argument from the user, which I have then assigned to a variable. I have nested while and for loops together. While loop helps me read the lines, and for loop helps me print asterisks for the number of times specified for each line. I have used numbers 20, 18, 5, 32 and 4 as it is in the example.




```
kirchhoff@WeirdGuyVery:~/ceng322hw1/question1$ ./star_printer.sh numbers.txt
********************
******************
*****
********************************
****
```

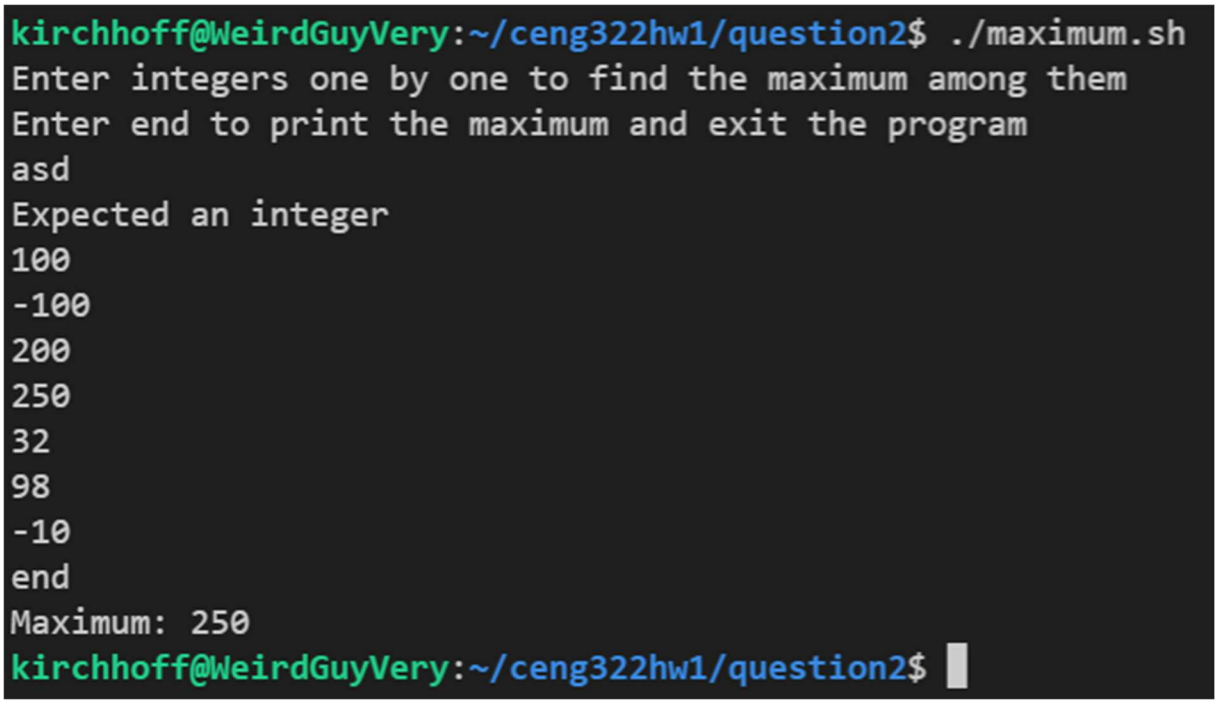
I assumed there is a new line at the end of the text file from which the program reads the integers. Otherwise, if I delete the new line at 6th row in text file, the program ignores the last line as shown:

```
kirchhoff@WeirdGuyVery:~/ceng322hw1/question1$ ./star_printer.sh numbers.txt
********************
******************
*****
********************************
kirchhoff@WeirdGuyVery:~/ceng322hw1/question1$ ./star_printer.sh numbers.txt
```

Question 2)

The program first checks whether the user ends the program directly after execution. If so, the program terminates with an exit code 1. I have chosen to terminate with exit code 1 because the program does not compare anything, which is its intended job. If user enters a number, it will be assigned as the first maximum number. I have used a while loop to see whether the arithmetic equivalent of the input is equal to itself to force the user to enter an integer, user will see an error message until a valid integer is given. The first input causes a problem if it is a string after assigning it as the maximum, therefore I have chosen such a method. Second and further inputs don't cause a problem.

In the next while loop, I chose the word "end" as the sentinel. The user can write as many integers as they want until "end" is given as an input. The program checks whether the inputs are greater than the previous maximum integer in the while loop. If so, the integer is assigned as the new maximum.



Question 3)

For the third question, I have branched two conditions both for which the user uses an argument while executing the program and for which no arguments are given. If no arguments are given, the program will work on the current working directory and delete all zero size files. If an argument is given, this time the zero size folders in the directory, which is given as an argument to the program, will be deleted unless it is an invalid path. If there is an invalid path given as the argument, program will prompt an error message. I have chosen to solve this problem using “find” because it already has a built-in -size option, which can be used check whether the files are zero length. “find” will return the list of files whose sizes are zero when the option is used as “-size 0” and they will be assigned to a variable in the for loop to be then deleted one by one. Also, I have used “-maxdepth 0” option to indicate that I don’t want the program to execute recursively. If I don’t do so, “find” will run recursively for sub-directories as well, removing zero files until the bottom. I have chosen to exit with code 1 if the directory does not exist when given as an argument.