Assignment name : get\_next\_line

Expected files : get\_next\_line.c get\_next\_line.h

Allowed functions : read, free, malloc

--------------------------------------------------------------------------------

Write a function named get\_next\_line which prototype should be:

char \*get\_next\_line(int fd);

Your function must return a line that has been read from the file descriptor

passed as parameter.

What we call a "line that has been read" is a succession of 0 to n characters

that end with '\n' (ascii code 0x0a) or with End Of File (EOF).

The line should be returned including the '\n' in case there is one at the end

of the line that has been read.

When you've reached the EOF, you must store the current buffer in a char \* and

return it. If the buffer is empty you must return NULL.

In case of error return NULL.

In case of not returning NULL, the pointer should be free-able.

Your program will be compiled with the flag -D BUFFER\_SIZE=xx, which has to be

used as the buffer size for the read calls in your functions.

Your function must be memory leak free.

When you've reached the EOF, your function should keep 0 memory allocated with

malloc except the line that has been returned.

Calling your function get\_next\_line in a loop will therefore allow you to read

the text available on a file descriptor one line at a time until the end of the

text, no matter the size of either the text or one of its lines.

Make sure that your function behaves well when it reads from a file, from the

standard output, from a redirection etc.

No call to another function will be done on the file descriptor between 2 calls

of get\_next\_line.

Finally we consider that get\_next\_line has an undefined behavior when reading

from a binary file.

You should use the test.sh to help you test your get\_next\_line.