If you choose to complete this assignment you could earn up to 105/100, basically 5 EC points.

Assignment Info:
Suppose Billy has a book of N pages. He wants to flip to a page in the book and wants to know the least number of page turns it will take him to reach the page he wishes to see. He can either start flipping to the desired page from the front of the book or the back of the book.

Assume, page 1 starts on the right page when the book is first opened. Page 2 is on the back of that page. Page 3 is on the right, across from page 2, and so on.

You will read the program input from a file. The input will be as follows, the first line is the number of total pages in the book and the second line is the page Billy wants to turn to.

For example, if the input is:

6
2

Your output should be:

1 page from the front
Note: Since the answer is 1, the word 'page' is singular.

Explanation:
There are 6 pages in this book. Since you are flipping to page 2, and it would be quicker to flip from the front, you only have to make 1 page turn.

Another example:
For input:
50
33

Output would be:
9 pages from the back

Code Requirements:

I will provide you with a functions.h file.
This file has the following function prototypes. You are **NOT** allowed to change the function prototypes for this assignment. The functions and descriptions are provided below.

void howManyTurns(int, int);

The parameters represent the total pages in the book and the page you want to turn to. This is where the real work for this assignment lives. As the name indicates this function will determine the least number of page turns needed to get to the desired page. It will also determine if the turns will be from the front or the back of the book. Remember if there is only 1 page turn then make sure the word page is singular. Once the needed information is determined you will print the appropriate output. See example output above. Remember the Mimir environment uses a diff command with the auto-grader. Therefore, make sure you have the spacing and wording correct.

void readData(FILE*, int*, int*);

This function will read two numbers from the file. The first number is the number of pages in the book. The second is the page Billy wants to turn to. You will use fscanf to read the data and store it in the parameters passed to the function. You will then call the function checkData.

void checkData(int, int);

This function will use assert to ensure the total pages in the book is larger than the page Billy desires to turn to. You cannot turn to page 55 in a book that only has 45 pages. You **MUST** use assert. If you have never heard of or used assert, no worries, it is simple. Look up assertion in C.

You are required to add header guards to functions.h file.

You MUST add comments above each function in the **functions.h** file. The comments must follow the following format:

Return: (if there is a return you are to explain what is being returned. If no return then put void.)

Parameters: (list the type of parameters and a short description of what it will be used for in the function.)

Description: (You are to write a detailed description of the functions purpose and how it will accomplish its purpose. In other words, describe what it is doing. It does not matter if the function is small and self-explanatory, you must provide a description.)

A substantial number of points will be deducted if you do not provide appropriate documentation.

I have also provided you with an empty functions.c file. You will implement the functions listed above in the functions.c file.

Lastly, I have provided you with driver.c which is where main() lives.
Since you will be reading data from a file you will need a file pointer. The name of the file to be read will be provided on the command line during execution (argv[1]). You are to use an assert statement to ensure the file opened successfully (make sure the file pointer is not NULL).
You will then call the function readData and howManyTurns.

Because leaving a file pointer unnecessarily open is a potential security risk, you must close the file pointer after you are finished with it.