Computer Science I Program #4: Scholarly Reader (Sorting)
Please check Webcourses/Mimir for the Due Date
Read all the pages before starting to write your code

Introduction: For this assignment you have to write a c program that will utilizes either merge sort or quick sort algorithm with some optimization.

What should you submit?

Write all the code in a single file and upload the .c file, leak detector c file, leak detector header file, and your test strategy (yourname_test.txt), and your generated three files test_in1.txt, test_in2.txt, test_in3.txt and the corresponding outputs test_out1.txt, test_out2.txt, and test_out3.txt. You need to submit all of them to mimir.

Please include the following commented lines in the beginning of your code to declare your authorship of the code:

/* COP 3502C Assignment 4

This program is written by: Your Full Name */

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Assignment rules mentioned in syllabus, are also applied in this submission. The TA and Instructor can call any students for explaining any part of the code in order to better assess your authorship and for further clarification if needed.

Caution!!!

Sharing this assignment description (fully or partly) as well as your code (fully or partly) to anyone/anywhere is a violation of the policy. I may report to office of student conduct and an investigation can easily trace the student who shared/posted it. Also, getting a part of code from anywhere will be considered as cheating.

Deadline:

See the deadline in Webcourses/mimir. No late submission will be accepted. An assignment submitted by email will not be graded and such emails will not be replied according to the course policy.

What to do if you need clarification on the problem?

I will also create a discussion thread in webcourses, and I highly encourage you to ask your question in the discussion board. Maybe many students might have same question as you. Also, other students can reply, and you might get your answer faster.

How to get help if you are stuck?

According to the course policy, all the helps should be taken during office hours. Occasionally, we might reply in email, but we cannot guarantee a timely response.

Problem Description

You want to prove to your parents that you are a scholar. Naturally, the more books you've read, the scholarly you must be. Unfortunately, you don't actually like reading too much and have a limit for the number of pages you are willing to read. Luckily, your parents aren't aware that some books have many more pages than other books! Thus, to impress your parents, you will try to read as many different books as possible, without going over your page limit!

The Problem

Given a list of the number of pages in all the books in the school library, and a maximum limit of the number of pages you are willing to read, determine the maximum number of books you can fully read!

The Input (to be read from in.txt file)

The first line will contain a single positive integer, c ($c \leq 25$), representing the number of test cases to process. The test cases follow.

The first line of each test case will contain a two space separated positive integers, n ($n \leq 100000$), representing the number of books in the school library, and L ($L \leq 10^{14}$), representing the maximum number of pages you are willing to read.

The second line of each test case will contain n space separated integers, each of which represents the number of pages in one of the n books in the school library. Each of these integers will be in between 1 and 10^9, inclusive.

The Output (to be printed to both standard console output as well as out.txt file)

For each input case, output the maximum number of different books you can read without going over your maximum page limit.

Sample Input

```
3
5 20
6 12 3 10 2
5 21
12 3 6 10 2
10 31
9 6 6 3 8 2 12 15 13 7
```

Sample Output

```
3
4
5
```

Implementation Restrictions/ Run-Time/Memory Restrictions

1. You must read the page values into a dynamically allocated array.
2. For full credit, your algorithm must run in O(nlgn) time **and** you must implement either your own **Merge Sort** or **Quick Sort**, coupled with an efficient method to solve the problem after sorting the input data.

3. For full credit, you must have appropriately designed functions. **In particular, any correct solution which is fully coded in main will NOT receive full credit.**

4. You must only declare your array variable **INSIDE** your case loop (see you have multiple cases in a single file).

5. You must free all dynamically allocated memory for full credit.

6. You must use the data type `long long` when necessary to avoid overflow (however, don't misuse long long for all variables. Read the input specification to determine this). (As a reminder, the maximum value that can be stored in the type `int` is roughly 2 billion while the maximum value that can be stored in the type `long long` is roughly 8 x 10^{18}. The percent code (aka format specifier) for the type `long long` is %lld.)

More explanation about the Deliverables

You must submit the following files in mimir:

1) A source file, *main.c*. Make sure to read from in.txt file in your code and produce standard console output as well as out.txt file in your code.

2) A file describing your testing strategy, *yourname_test.txt*. This document discusses your strategy to create test cases to ensure that your program is working correctly. A good idea would be writing a code to generate test cases based on the input specification to test whether your code can deal with large numbers specified in the input speicification. If you used code to create your test cases, just describe at a high level, what your code does, no need to include it.

3) Files test_in1.txt, test_in2.txt, test_in3.txt and the corresponding outputs test_out1.txt, test_out2.txt, and test_out3.txt. **(Note: At least one test case MUST BE generated by a program and NOT by hand of size 10^5 and one test case must have a maximum page limit greater than 10^{10} for full credit here.)**

4) The leak detector c file, leak detector header file

Little hints:

Read how many test cases you have in the file and then for each test case, allocate array for n size and fill up the array.
The naïve approach without any optimization would be pretty straight forward. You just need to sort the books based on pages and then sum up the pages of the books you are able to read based on the page limit and print the book count.