Computer Science I Program #4: Scholarly Reader (Sorting)

Please check Webcourses for the Due Date

Read all the pages before starting to write your code

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 standard input)

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 standard out)

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.

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. (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 8×10^{18}. The percent code for the type `long long` is %lld.)

Deliverables

You must submit one file over WebCourses:

1) A source file, *scholar.c*. Please use stdin, stdout. There will be an automatic 10% deduction if you read input from a file or write output to a file.

2) A file describing your testing strategy, *lastname_Testing.doc(x) or lastname_Testing.pdf*. This document discusses your strategy to create test cases to ensure that your program is working correctly. 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 *scholar.in* and *scholar.out*, storing both the test cases you created AND the corresponding answers, respectively. **(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.)**