Business Buyout

Objective
Give practice with sorting in C
Give additional practice with stacks in C

Story
The Benny Travis Shop (BTS) (a rival to your place of work–KAT Pet Shop) is working on grabbing a share of your customers. They are doing so by buying out some local pet shops to accumulate enough wealth to corner the pet shop market. For this problem you can assume that a shop can buyout another shop if the shop doing the purchase has strictly more wealth than the shop being purchased. When a shop is bought the result is a shop that has a combined wealth of the original 2 shops.

If BTS gets too big there will be nothing the other shops can do to prevent them from causing all the other shops to go out of business. You will assume no other shops besides BTS will be buyouting competitors.

Problem
You will be given the initial wealth of BTS and the wealth of a number of shops. You need to determine the least amount of shops needed to be bought out such that BTS will be more wealthy than any other shop.**DO NOT USE BUILT-IN QSORT.**

Input
The input will begin with a line containing 2 integers, N and W ($1 \leq N \leq 500{,}000$; $1 \leq W \leq 1{,}000{,}000{,}000$), representing the number of competing pet shops that BTS can purchase the starting wealth of BTS respectively. Following this will be N lines each containing a single integer, V ($1 \leq V \leq 1{,}000{,}000{,}000{,}000$), representing the wealth of a competing pet shop.

Output
Print 1 integer representing the least number of shops that BTS needs to buyout such that they have the highest wealth pet shop. If BTS cannot buy enough shops to become the wealthiest, output -1 instead.

By Kaylee :]

Sample Input	Sample Output
6 5 6 3 5 10 4 9	2
4 10 1 2 3 20	-1

Sample Explanation

FOR CASE 1

You have 6 shops and BTS has a starting wealth of 5. They can purchase the second shop at a value of 3. The shop's wealth becomes 5 + 3 = 8. After this they could purchase the 1st shop at a value of 6. The shops wealth then becomes 8 + 6 = 14. It takes only 2 purchases to become wealthiest.

FOR CASE 2

The most wealth the shop can acquire is 16 by purchasing the first 3 shops. At that point BTS would no longer be able to buyout any other shop. This means they cannot become the wealthiest. For this reason -1 is the result for the case.

Hints

To-Purchase Stack: Track a stack of possible shops that can be bought.

Sort: You should sort the competitor shop wealths from least to greatest so that you know which shops BTS can buy.

Nlog(N) Sort: To not run slow you must use either quick sort or merge sort,

DO NOT USE BUILT-IN QSORT.

Track Shop: Depending on your wealth you will be able to purchase other stores. If the stores are sorted by wealth then a prefix (contiguous subarray starting at the beginning) of the stores will be purchasable, and a suffix (contiguous subarray ending at the end) of the stores will be too expensive. The turning point between purchasable and too expensive should be stored to prevent needlessly comparing values that have already been added.

Simulate Buyout: When a competitor is bought out the wealth of the company changes. This can mean additional shops may be purchased. Start from the last shop you were not able to add to the To-Purchase collection, and add shops while you can purchase these yet-to-be-added shops.

It is optimal to purchase the most wealthy purchasable shops first. The stack will ensure that you purchase the wealthiest shops first if you add them in order from least wealthy to most.

Grading Criteria

- Read/Write from/to **standard** input/output (e.g. scanf/printf and no FILE *)
    - 5 points
- Good comments, whitespace, and variable names
    - 15 points
- No extra input output (e.g. input prompts, “Please enter the number of cats:”)
    - 5 points
- Read the shops into an array
    - 5 points
- Sort the shops by their buyout value
    - 5 points
- Store the shops that can be bought in a stack ordered from greatest value to least
    - 5 points
- Store an index into the sorted array of shops that determines which ones have been added to the stack
    - 5 points
- Simulate by having BTS buyout one shop at a time
    - 5 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using “gcc -std=gnu11 -lm”.

Sometimes a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem implement your own sort function. ***Without this programs will earn at most 50 points! Using qsort will not be allowed.***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.