CS Data Structure

Fall 2018

Program Exercise #6

Last update: 2018/011/15

*Due Date: 2018/12/11 (Tue.) 23:59:00

Problem: Minimum addition cost

Again, this question is a modified version from an ACM problem. Assume that you have got some numbers (addends) at your hand and you want to add them up, two numbers at a time, with a minimal sum of addition cost. The addition cost is the sum of the addition. For example, the cost for adding 3 and 7 is 10, and the cost for adding 12 and 4 is 16, etc. If you want to achieve the minimal sum of addition cost, you should choose the smallest two numbers in each round and add them up. Let's use `{12, 4, 7, 3, 10}` as an example:

Available Numbers	Addition	Cost	Total Cost
12, 4, 7, 3, 10	3 + 4 = 7	7	7
12, 7, 10, 7	7 + 7 = 14	14	21
12, 10, 14	10 + 12 = 22	22	43
14, 22	14 + 22 = 36	36	79

Thus, the minimum addition cost is 79. You will need to keep adding numbers until there is left with only one number. If only one addend is given, no addition is required and thus the addition cost is 0.

Input

The input file (a text file) contains multiple test cases. Each line in the file describes a single test case. The first integer represents the number of addends in the current test case. The rest of the numbers are the addends of the test case. Thus, the example given above can be represented as:

```
5 12 4 7 3 10
```

All addends are positive integers. Each test case contains at least one addend. The number of addends in a test case is not limited.

The input file name must be entered from the command-line as an argument like this:

```
myHomework5.exe myInput.txt
```

Output

You are asked to compute and output minimum addition cost for each test case. Make sure that your output conforms to the following requirements:

- Output all results on screen.
- The solution of each test case occupies exactly one line.

Sample Input

```
5 12 4 7 3 10
8 22 4 5 16 11 9 17 9
1 3
19 11 32 3 7 20 12 2 9 14 25 8 31 19 24 16 6 29 26 15
```

Sample Output

```
79
266
0
1247
```

Other Requirements

- You are required to implement your solution via **a min heap**.
- There is no limit on the number of addends in each test case. Please construct your heap by dynamic memory allocation. **You are not allowed to use any automatic allocation/deallocation class/library such as `vector` in C++, and you are not allowed to use fixed memory allocation**.
- Make sure that your program is compilable. This means:
  - You are asked to upload your .c or .cpp file instead of sending us your project files or other types of files.
  - If you happen to use some of the libraries from C++, make sure your file extension is cpp not c. A .c program calling C++ libraries is NOT compilable.
  - If your program turns out to be incorrect or uncompilable, you may write a text file report, describing how far you have gone to, your implementation idea, and the problem you have encountered.
- Please check if your program has double file extensions. A file like this (e.g. M06455505_program5.cpp.c) **will be graded zero**.
- Comments and header block are required. We do not restrict the format of your comments, but you should have at least the following:
  - Program name, student name, student ID, purpose of the program in your header block.
  - The purpose of each variable (short description only, no need to write a paragraph for each variable), unless you are positively sure that the variable name is self-explanatory.
  - The purpose of each function and its input and output.
  - Short description for each key step.
- Consistent indentation and spacing makes your program readable. We do not restrict your program to any indentation and spacing format. But if you have no indentation at all, your grade will be deducted.
- **You SHALL NOT copy your entire source code or part of it from your friends or other resources on the web.**
- **If any copied program is caught, both persons will be given zero.**
- **Any late submission will be given zero.**