Задача 2. SQSORT

Тъй като Радо не обича да пише условия за задачи, това ще е сравнително кратко. Задачата е **интерактивна**, като системата има редица от числа $A_0, A_1, \ldots, A_{N-1}$. За жалост, Вие знаете само дължината на тази редица, но не и стойностите на елементите ѝ. Целта е да сортирате **ненаредените двойки** позиции, така че сумите на съответстващите им елементи да са в **ненамаляващ** ред. Формално, искаме да намерим редица $(i_0, j_0), (i_1, j_1), \ldots, (i_{\frac{N(N+1)}{2}-1}, j_{\frac{N(N+1)}{2}-1})$, такава че:

1) За всички $0 \leq i' \leq j' < N$, съществува позиция $0 \leq k < \frac{N(N+1)}{2}$, такава че $(i', j') = (i_k, j_k)$.

2) $A_{i_{k-1}} + A_{j_{k-1}} \leq A_{i_k} + A_{j_k}$ е изпълнено за всяко $1 \leq k < \frac{N(N+1)}{2}$.

Тъй като не знаете стойностите в масива A, ще можете да питате въпроси за сравняване на сумите на две двойки от позиции. По-формално, за $0 \leq a, b, c, d < N$, ще може да питате въпроса дали $A_a + A_b < A_c + A_d$ е изпълнено. Искате да зададете колкото се може по-малко въпроси.

Детайли по имплементацията

Вашата програма трябва да имплементира функцията solve, която ще бъде извикана точно веднъж и трябва да върне редицата от сортираните по сума двойки. Прототипът е трябва да е:

```
std::vector<std::pair<int, int>> solve(int n);
```

Вашата програма може да вика функцията cmp, която има следния прототип:

```
bool cmp(int a, int b, int c, int d);
```

Едно извикване на тази функция ще наричаме *заявка* и тя ще връща true, ако $A_a + A_b < A_c + A_d$, и false в противен случай. Условието $0 \leq a, b, c, d < N$, трябва да е изпълнено за всички заявки на вашата програма.

Напишете програма **sqsort.cpp**, която имплементира функцията `solve` и не съдържа `main` функция. Също така, не трябва да четете от стандартния вход или да пишете на стандартния изход. За успешно компилиране, трябва и да добавите хедър файла **sqsort.h** чрез указване към предпроцесора:

```
#include "sqsort.h"
```

Стига тези условия да са изпълнени, Вашата програма може да съдържа всякакви помощни функции,класове, променливи и т.н.

Ограничения

$100 \leq N \leq 2000$

Оценяване

Всеки тест се оценява поотделно, като точките, които ще получите за него, зависят от броя заявки Q, които програма Ви е направила преди да върне сортираната редица:

• 0 точки, ако $Q > 5 \times 10^7$ или ако редицата върната от `solve` не е сортирана по гореописания начин

• $min\left(1, \left(\frac{2N^2+1}{Q+1}\right)^{1.25}\right) \times P$, където P е максималния възможен резултат за съответния тест, в противен случай

Локално тестване

Предоставени са Ви файловете **sqsort.h** и **Lgrader.cpp**, които трябва да компилирате заедно с програмата си. При стартиране, програмата ще прочете цялото число N, последвано от целите числа $A_0, A_1, \ldots, A_{N-1}$. След това, тя ще извика функцията solve и ще отговаря на Вашите cmp заявки. Като изход, грейдърът ще отпечата броя заявки, които вашата програма е използвала, или съобщение, че редицата не е била сортирана. Свободни сте да променяте тези файлове по какъвто начин си искате. Гарантирано е, че грейдърът на журито се държи еквивалентно на **Lgrader.cpp** и няма да се адаптира към заявките Ви.

Примерна комуникация

№	Действия на `sqsort`	Действия и отговори на журито
1.		`solve(3)`
2.	`cmp(0, 1, 0, 1)`	`return true`
3.	`cmp(0, 2, 0, 0)`	`return true`
4.	`cmp(0, 2, 0, 1)`	`return false`
5.	`cmp(1, 2, 1, 1)`	`return false`
6.	`cmp(2, 2, 1, 2)`	`return false`
7.	`cmp(1, 1, 0, 1)`	`return true`
8.	`cmp(1, 2, 0, 1)`	`return true`
9.	`cmp(2, 2, 0, 1)`	`return false`
10.	`cmp(2, 2, 0, 2)`	`return true`
11.	`return {(1, 1), (1, 2), (0, 1), (2, 2), (0, 2), (0, 0)}`	

Обяснение на примерната комуникация

1. $N = 3$. Редицата A е равна на $\{5, 1, 3\}$, но функцията `solve` не знае това.
2. $A_0 + A_1 = 6 < 10 = A_0 + A_0$
3. $A_0 + A_2 = 8 < 10 = A_0 + A_0$
4. $A_0 + A_2 = 8 \geq 6 = A_0 + A_1$
5. $A_1 + A_2 = 4 \geq 2 = A_1 + A_1$
6. $A_2 + A_2 = 6 \geq 4 = A_1 + A_2$
7. $A_1 + A_1 = 2 < 6 = A_0 + A_1$
8. $A_1 + A_2 = 4 < 6 = A_0 + A_1$
9. $A_2 + A_2 = 6 \geq 6 = A_0 + A_1$
10. $A_2 + A_2 = 6 < 8 = A_0 + A_2$
11. $A_1 + A_1 \leq A_1 + A_2 \leq A_0 + A_1 \leq A_2 + A_2 \leq A_0 + A_2 \leq A_0 + A_0$

Програмата е използвала 9 заявки и успешно е сортирала редицата.