КОНТРОЛНО ПОДБОРНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
25-26 юли 2020 г., Група А

Задача АК6. МУЛТИСОРТ

Хъмпти-Дъмпти седеше на стената,
Карта след карта падна на земята.
Всички царски хора и царските коне
не могат вече да ги сортират по местата им, не!

Както стихотворението казва, Хъмпти-Дъмпти изтървал тестето си от N с карти и сега никой не може да го нареди в правилния ред. Затова той помолил Алиса за помощ. Тя е чувала, че има бързи методи за сортиране, които се базират на сравняване на елементите по двойки, но това ѝ се струва доста неефикасно. Все пак тя може да хване до K карти в ръката си и веднага да знае в какъв ред трябва да са те. Чуди се дали няма някой метод за сортиране, който може да се възползва от това нейно умение. Нейната цел е да минимизира броя такива K-странни сравнения, които ще извърши, за да подреди тестето.

След дълго мислене, тя не се сетила за добър метод. Затова сега тя пита Вас за помощ. Помогнете ѝ като напишете програма **multisort.cpp**, която да комуникира с Алиса (програмата на журито), за да сортира картите с колкото се може по-малко сравнения.

Детайли по имплементацията

Вашата функция `multisort` трябва да има следния прототип:

```
std::vector<int> multisort(int n, int k);
```

Тя ще бъде извикана точно веднъж и ще получи като аргументи броя карти за сортиране и броя карти, които Алиса може да сравни наведнъж. Функцията трябва да върне вектор, който съдържа пермутация на числата от 0 до $N - 1$. Това е пермутацията, в която трябва да се наредят картите, така че те да са подредени в нарастващ ред. Картите определяме с техните индекси, т.е. позициите им в началното разбъркване.

Функция `compare` на журито има следния прототип:

```
void compare(std::vector<int>& elems);
```

Вашата програма може да я вика колкото си пъти иска. Като аргумент ѝ се дава референция към вектор съдържащ индексите на картите, които да бъдат сравнени от Алиса. Векторът трябва да съдържа до K валидни индекса на карти, но е позволено да съдържа повторения. Функцията ще сортира вектора, така че указаните карти да са в нарастващ ред. Това ще се случи със сложност $O(M \log M)$, където M е дължината на вектора.

Вашата програма трябва да имплементира функцията `multisort`, но не трябва да съдържа функция `main`. Освен това, тя не трябва да чете от стандартния вход или да печата на стандартния изход. Програмата Ви също така трябва да включва хедър файла **multisort.h** чрез указание към препроцесора:

```
#include "multisort.h"
```

Стига да спазва тези условия, програмата Ви може да съдържа каквито и да е помощни функции, променливи, константи и прочие.

КОНТРОЛНО ПОДБОРНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
25-26 юли 2020 г., Група А

Ограничения

$N = 20\,000$ във всички тестове
$2 \leq K \leq 1000$

Оценяване

Всеки тест се оценява поотделно. За даден тест Вашето решение ще получи точки, различни от 0, ако функцията `multisort` успешно приключи изпълнение и върне вектор с дължина N, който съдържа правилно сортираната пермутация от индекси на карти. Точките, които ще получите на теста са равни на максималният брой точки за теста, умножени по:

$$\min\left(1, \left(\frac{authorScore + 1}{yourScore + 1}\right)^{0.75}\right)$$

Тук $yourScore$ е броя сравнения, които е използвало вашето решение, а $authorScore$ е броя сравнения, които е използвало авторовото решение в най-лошия от 10 опита. По-точно, авторовото решение е пуснато на 30 теста с различни пермутации, но с еднакви N и K (един от тези тестове е този, на който Вашето решение бива тествано). След това за $authorScore$ е взет най-лошият (максималният) от десетте резултата, които авторовото решение е постигнало.

Очакваният брой сравнения, които авторовото решение ще използва за дадени N и K като сложност е равен на теоретичния минимум (получен, като се разглежда колко информация дава едно сравнение). По-конкретно, за не твърде малки стойности на K авторовото решение използва около два пъти повече сравнения от теоретичния минимум.

Тестове

Процент тестове	K
10%	$2 \leq K < 5$
10%	$5 \leq K < 10$
10%	$10 \leq K < 20$
30%	$20 \leq K < 100$
40%	$100 \leq K \leq 1000$

Пермутацията на картите в даден тест ще бъде произволно генерирана, така че всяка пермутация има равен шанс да се падне.

Локално тестване

Предоставени Ви са файловете **multisort.h** и **Lgrader.cpp**, които можете да компилирате заедно с Вашата програма, за да я тествате.

При стартиране на програмата трябва да се въведат K както и сийд за генериране на пермутацията. Ако искате да я конфигурирате по друг начин, може да правите каквито си промени искате по предоставените Ви файлове.

Гарантирано е, че грейдърът в системата ще се държи както предоставения Ви локален грейдър. Най-съществено, имплементацията на функцията `compare` ще е същата.

КОНТРОЛНО ПОДБОРНО СЪСТЕЗАНИЕ
НА РАЗШИРЕНИЯ НАЦИОНАЛЕН ОТБОР
25-26 юли 2020 г., Група А

Примерна комуникация

№	Действия на `multisort`	Действия и отговори на журито
1.		`multisort(4, 3)`
2.	`compare({0, 1, 2})`	`elems = {1, 0, 2}`
3.	`compare({1, 2, 3})`	`elems = {1, 3, 2}`
4.	`compare({0, 3})`	`elems = {3, 0}`
5.	`return {1, 3, 0, 2}`	

Обяснение

$N = 4$ (с цел демонстрация) и $K = 3$.
Тук стойностите на четирите карти са: $v_0 = 2, v_1 = 0, v_2 = 3, v_3 = 1$.
Първото сравнение открива, че: $v_1 < v_0 < v_2$.
Второто сравнение открива, че: $v_1 < v_3 < v_2$.
Третото сравнение открива, че: $v_3 < v_0$.
Единственото възможно решение е: $v_1 < v_3 < v_0 < v_2$.
Програмата на състезателя общо е използвала 3 сравнения.