서문
지루한 알고리즘 수업을 가르치는 다소 멍청한 김 교수는 매 학기마다 학생들의 기말 점수를 기반으로 최적의 성적을 어떻게 줄지 고민해왔지만, 아직 좋은 방법을 찾지 못했다.
짜증나는 Chat-GPT 덕분(?)에 과제가 없었던 김 교수는, 이 문제를 숙제로 내면 일석이조가 되겠다고 생각해버렸다. (젠장! 또 이딴 일이 벌어지다니!)
이제 이 과제를 통해 김 교수를 도와주자. 김 교수가 고마워할지는 아무도 모른다. 지금까지 김 교수는 점수 차이가 크게 나는 구간을 수동으로 찾아 그 구간을 성적 구분 기준점으로 사용해왔다.

method 1

`n`명의 학생 점수를 입력으로 받아 내림차순으로 정렬한 뒤, 이들을 k개의 그룹으로 나눈다.
그룹 번호가 낮을수록 점수가 높은 그룹이라고 가정한다.
`i = 1`부터 `k - 1`까지 각 그룹에 대해, `i`번째 그룹의 **최소 점수**와
`i + 1`번째 그룹의 **최대 점수**의 차이를 계산하고, 이 차이들의 총합을 **최대화**한다.
단, **각 그룹에는 반드시 최소한 `1`명의 학생이 포함되어야 한다.**

method 2

`n`명의 학생 점수를 입력으로 받아 정렬한 후, 이들을 `k`개의 그룹으로 나눈다.
역시 그룹 번호가 낮을수록 점수가 높은 그룹이라고 가정한다.
각 그룹 내 학생들의 점수가 **서로 비슷**하면 이상적일 것이다.
따라서, 각 그룹의 점수 분산을 계산하고, **모든 그룹의 분산의 합을 최소화**하는 것이 목표이다.
단, **각 그룹에는 최소한 한 명 이상의 학생이 포함되어야 하며**, 한 명만 있는 그룹의 분산은 **`0`으로 간주**한다.

입력

표준 입력(Standard Input)에서 입력을 받는다.
첫 번째 줄에는 **정수** `n`과 `k`가 주어진다.
두 번째 줄에는 `n`명의 학생 점수가 주어지며, **학생 번호 순(즉, 학번 순)**으로 정렬되어 있다.
여기서 `n`은 `k` **이상인 수**이다.
각 점수는 **`0` 이상 `1000` 이하**의 정수이며, **동점자가 있을 수 있다.**
놀랍게도 `n`은 **최대 `10000`**까지 가능하다.
`k`는 **`1` 이상 `12` 이하**의 정수이다.
(참고: 어떤 학교에서는 각 성적 등급(A, B, C, D)을 2~3개의 세부 등급으로 나누기도 한다.)

출력

표준 출력과 파일에 **동시에** 출력한다.
첫 번째 줄에는 **방법 1(Method 1)**의 **(최대) 점수 차이 합**을 표준 출력에 출력한다.
방법 1의 그룹 구성 결과는 **`Partition1.txt` 파일**에 저장한다.

- 이 파일의 첫 번째 줄은 **`1`번 그룹**, `k`번째 줄은 **`k`번 그룹**에 해당한다.
- 각 줄에는 해당 그룹에 속한 학생들을 `"학생 번호 (학생 점수)"` 형식으로 나열하며, **학생 번호 오름차순**으로 정렬한다.

두 번째 줄에는 **방법 2(Method 2)**의 **(최소) 분산의 합**을 표준 출력에 출력하며, **소수점 셋째 자리까지 반올림**하여 표시한다.
방법 2의 그룹 구성 결과는 **`Partition2.txt` 파일**에 저장한다.

- 이 파일도 첫 번째 줄이 1번 그룹, k번째 줄이 k번 그룹에 해당하며,
- `"학생 번호 (학생 점수)"` 형식으로 각 줄에 **학생 번호 오름차순**으로 나열한다.

```
Example
Input (standard/console input)
15 3
50 50 10 20 50 10 50 50 20 20 50 50 50 50 10
Output (standard/console output)
40
0
Output (filename: Partition1.txt)
1(50) 2(50) 5(50) 7(50) 8(50) 11(50) 12(50) 13(50) 14(50)
4(20) 9(20) 10(20)
3(10) 6(10) 15(10)
Output (filename: Partition2.txt)
1(50) 2(50) 5(50) 7(50) 8(50) 11(50) 12(50) 13(50) 14(50)
4(20) 9(20) 10(20)
3(10) 6(10) 15(10)

Input (standard/console input)
15 3
50 85 10 35 45 15 75 80 25 30 55 60 65 70 5
Output (standard/console output)
20
197.917
Output (filename: Partition1.txt)
1(50) 2(85) 5(45) 7(75) 8(80) 11(55) 12(60) 13(65) 14(70)
4(35) 9(25) 10(30)
3(10) 6(15) 15(5)
Output (filename: Partition2.txt)
2(85) 7(75) 8(80) 13(65) 14(70)
1(50) 5(45) 11(55) 12(60)
3(10) 4(35) 6(15) 9(25) 10(30) 15(5)
```

논의 (Discussion)

아래의 각 제약 조건에 대해, **접근 방식이 어떻게 달라지는지** 를 보고서에 **자세히 기술**

** 1. 동일한 점수를 가진 학생들은 반드시 같은 그룹에 속해야 한다.**

이 조건은 성적 부여의 **공정성** 을 반영,
동일한 점수를 받은 학생들을 자의적으로 나누는 것을 방지 하기 위해 추가되었다.
동일한 점수를 가진 학생들은 그룹 분할 시 구분이 불가능한 동일 단위로 간주되며
그룹을 나눌 때 **하나의 묶음으로 취급되어야 한다.**

** 2. 그룹별 학생 수에 대한 제한**

예를 들어, `1`**번 및** `2`**번 그룹의 학생 수 총합이 전체 학생 수(**`n`**)의** `30%`**를 초과할 수 없다.**
또한, `1~4`**번 그룹의 학생 수 총합이 전체의** `70%`**를 초과할 수 없다.**
학생들은 이와 같은 누적 인원 제한 조건을 **위반하지 않도록** 할당되어야 한다.
알고리즘이 이러한 조건을 만족하는 구성을 **찾지 못할 경우**, **불가능함을 감지하고 보고해야 한다.**

** 2. 그룹 내 점수 범위 제한**

예를 들어, **각 그룹 내 최고점과 최저점의 차이가 주어진 임계값 R을 초과할 수 없다.**
이 제약 조건은 **점수 차이가 큰 학생들이 같은 그룹에 배정되는 것을 방지**하기 위한 것이다.
조건이 추가될 경우, **알고리즘이 어떻게 조정되어 모든 그룹이 이 조건을 만족하도록 만드는지 설명해야 한다.**

** 2. 우선순위 기반 그룹 구성**

각 학생은 **우선순위 값** p_i 를 가진다.
우선순위의 총합이 높은 그룹이 더 낮은 그룹 번호(즉, 더 좋은 등급)를 가져야 한다.
이러한 조건을 **그룹핑 전략에서 어떻게 표현하고 강제할 것인지**에 대해 논의하시오.