**A. "Все, Степан! Ти мене дістав!"**

|  |  |
| --- | --- |
| **Input file name:** | bubble.in |
| **Output file name:** | bubble.out |
| **Time limit:** | 100 ms |
| **Memory limit:** | 256 M |

Степан нещодавно відпочивав у Японії і привіз звідти нову жувальну гумку. На першій парі в університеті він поділився гумкою зі своїм товаришем. Дочекавшись моменту, коли лектор повернувся до дошки, на рахунок "три - чотири" хлопці дружньо почали надувати бульбашки. Відомо, що Степан надуває бульбашку до максимально можливого розміру за час ***t1***, після чого бульбашка миттєво лопається, і Степан починає надувати бульбашку заново з тією ж швидкістю. Товариш Степана робе те ж саме за час ***t2***.

Весь цей час викладач настільки захоплений доведенням теореми, що взагалі нічого не чує. І тільки коли обидві бульбашки лопнуть одночасно, викладач почує шум і обернеться. І тоді вже точно студентам попаде на горіхи, а більше усього тому, хто приніс на пару жувальні гумки.

Визначте, скільки часу хлопці можуть насолоджуватись надуванням бульбашок, не замічені викладачем.

Наприклад, якщо ***t1 = 2, t2 = 3***, то буде відбуватись наступне:

Степан надуває бульбашку з моменту часу ***t = 0*** до моменту часу ***t = 2***, потім бульбашка лопається, і він надуває бульбашку заново - з моменту часу ***t = 2*** до моменту часу ***t = 4***, а потім ще раз - з моменту часу ***t = 4*** до ***t = 6***.

Товариш Степана надуває бульбашку з ***t = 0*** до ***t = 3*** і ще раз з ***t = 3*** до ***t = 6***.

В момент часу ***t = 6*** бульбашки лопаються одночасно в обох студентів, викладач повертається і каже: "Все, Степан! Ти мене дістав!".

**Формат вхідних даних:** Перший рядок вхідного файлу містить два цілих числа ***t1, t2 (1 ≤ t1, t2 ≤ 109)***.

**Формат вихідних даних:** Вихідний файл повинен містити одне ціле число - час, протягом якого Степан з товаришем можуть насолоджуватись надуванням бульбашок.

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **bubble.in** | **bubble.out** |
| 2 3 | 6 |
| 1 16 | 16 |

#include<fstream>  
using namespace std;  
ifstream in("bubble.in");  
ofstream out("bubble.out");  
int gcd(int a,int b)  
{  
        while (b)  
        {  
                a%=b;  
                swap(a,b);  
        }  
        return a;  
}  
long long a,b;  
int main()  
{  
        in>>a>>b;  
        out<<a/gcd(a,b)\*b<<endl;

**B. Степан - бізнесмен**

|  |  |
| --- | --- |
| **Input file name:** | businessman.in |
| **Output file name:** | businessman.out |
| **Time limit:** | 300 ms |
| **Memory limit:** | 256 M |

Ужляндія, як відомо, країна з розвиненими торговими відносинами.

Степан вирішив спробувати зайнятися торгівлею і підзаробити собі на відпустку продажем комп'ютерної техніки. Для цього йому необхідно закуповувати техніку у інших продавців. Перш ніж почати роботу, він вирішив постежити за подіями на ринку Ужляндії і придумати, як отримувати найбільший прибуток.

Степан дізнався, що кожен продавець продає один комп'ютер, і кожен покупець готовий купити рівно один комп'ютер. Всього на ринку торгують ***N*** продавців, вартість комп'ютера у ***i***-го з них дорівнює ***Ai*** Ужляндських монет, причому ціни можуть відрізнятися у різних продавців. Крім того, він знайшов для себе ***M*** потенційних покупців, кожен з яких хоче купити комп'ютер за ***Bi*** монет. При цьому сам Степан може купити і продати будь-яку кількість комп'ютерів.

Степану необхідно отримати найбільший прибуток (вигідно купити і вигідно продати). Тому він звернувся за допомогою до Вас - кращому програмісту Ужляндії.

**Формат вхідних даних:** Перший рядок вхідного файлу містить розділення одиночним пропуском два цілих числа ***N, M (1 ≤ N, M ≤ 105)*** - кількість продавців на ринку Байтландіі і кількість потенційних покупців відповідно.  
Другий рядок містить ***N*** цілих чисел ***Ai (0 ≤ Ai ≤ 109)*** - вартості, за якими продавці готові продавати комп'ютери.  
Третій рядок містить ***M*** цілих чисел ***Bi (0 ≤ Bi ≤ 109)*** - суми, які потенційні покупці готові віддати при покупці комп'ютера.

**Формат вихідних даних:** Перший рядок вихідного файлу має містити одне ціле число ***S*** - розмір максимальної вигоди в монетах, яку може отримати Степан.

***Пояснення до прикладів:***

У першому прикладі Степан купує комп'ютери у 1-го і 2-го продавців і продає їх будь-яким двом покупцям.  
У другому прикладі Степану найбільш вигідно купити комп'ютери у 1-го, 4-го і 6-го продавців і продати 3-му, 4-му і 5-му покупцям.

***Оцінювання:***

***N + M ≤ 20***, для усіх ***і***: ***Ai, Bj ≤ 1000*** – не менше 40 балів

***N + M ≤ 2000***, для усіх ***і***: ***Ai, Bj≤ 1000*** – не менше 50 балів

***N + M ≤ 2000*** – не менше 75 балів

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **businessman.in** | **businessman.out** |
| 2 3  1 1  3 3 3 | 4 |
| 6 5  5 10 8 4 7 2  3 1 11 18 9 | 27 |

include<fstream>  
#include<vector>  
#include<algorithm>  
using namespace std;  
ifstream in("businessman.in");  
ofstream out("businessman.out");  
vector<long long> a,b;  
long long n,m,ans,x;  
int main()  
{  
        in>>n>>m;  
        for (int i=0;i<n;i++)  
        {  
                in>>x;  
                a.push\_back(x);  
        }  
        for (int i=0;i<m;i++)  
        {  
                in>>x;  
                b.push\_back(x);  
        }  
        sort(a.begin(),a.end());  
        sort(b.begin(),b.end());  
        for (int i=0;i<min(n,m);i++)  
        {  
                long long temp=b[m-(i+1)]-a[i];  
                if (temp>0) ans+=temp; else break;  
        }  
        out<<ans<<endl;  
}

**C. Transit**

|  |  |
| --- | --- |
| **Input file name:** | transit.in |
| **Output file name:** | transit.out |
| **Time limit:** | 1000 ms |
| **Memory limit:** | 256 M |

Країна Ужляндія має вигідне географічне розташування – її територія знаходиться на перетині важливих торгівельних шляхів. Одним із таких є торгівельний шлях, яким сусідня братська держава доставляє свої унікальні обігрівачі до інших країн.

На кордоні Ужляндії та братської держави, де починається цей шлях, розташований спеціальний пропускний пункт, через який щодня проїжджає потяг із величезною кількістю обігрівачів. Зовсім недавно між урядами двох братських країн було погоджено нові правила транзиту обігрівачів через територію Ужляндії на найближчі ***N*** днів. Згідно з новим договором має обратися певне число ***m*** – максимальна кількість обігрівачів в одному потязі. Тоді з кожного потяга, що транспортує ***Ai*** обігрівачів, буде відвантажено рівно ***Ai-m*** одиниць іноземної продукції (звичайно, якщо ***Ai > m***, інакше ж потяг рухатиметься без зупинок і нічого відвантажено не буде). Власне це й буде плата за проходження потяга територією Ужляндії, вона еквівалентна грошовим витратам на утримання залізничних колій. Сумарна кількість відвантажених в Ужляндії за ***N*** днів обігрівачів повинна бути не менша ***K***, інакше країна зазнає збитків.

Стала відома кількість обігрівачів у потязі в кожен із ***N*** днів (ця інформація надається за умовами контракту). Знайдіть максимальне число ***m***, при якому Ужляндія не зазнає економічних збитків.

**Формат вхідних даних:** В першому рядку записано два числа ***N, K (1 ≤ N ≤ 106, 1 ≤ K ≤ 2\*109)***. В наступному рядку задано ***N*** чисел – кількість обігрівачів у потязі в кожен з ***N*** днів, що не перевищує 109.

**Формат вихідних даних:** В єдиному рядку виведіть одне число – відповідь на задачу, гарантується, що відповідь завжди існує.

***Пояснення до прикладу:***

Всього територією Ужляндії пройде 4 потяги з 11, 6, 1 та 8 обігрівачами відповідно. Щоби країна не знала збитків, потрібно відвантажити не менше 7 обігрівачів. Очевидно, що максимальне можливе ***m***, яке задовольнить цю умову, буде рівне 6, тоді з потягів буде відвантажено відповідно 5, 0, 0, 2 обігрівачів, що в сумі дорівнює 7 і задовольняє умову.

***Оцінювання:***

***N ≤ 300*** – не менше 20 балів

***N ≤ 10000*** – не менше 40 балів

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **transit.in** | **transit.out** |
| 4 7  11 6 1 8 | 6 |

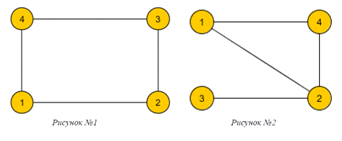
#include<fstream>  
#include<vector>  
#include<algorithm>  
using namespace std;  
ifstream in("transit.in");  
ofstream out("transit.out");  
vector<long long> a;  
long long n,k,ans,x,j;  
int main()  
{  
        in>>n>>k;  
        for (int i=0;i<n;i++)  
        {  
                in>>x;  
                a.push\_back(x);  
        }  
        sort(a.begin(),a.end());  
        x=0;  
        for (int i=n-1;i>=0;i--)  
        {  
                j++;  
                x+=a[i];  
                ans=max(ans,   (  (x-k)/j   )  );  
        }  
        out<<ans<<endl;  
}

**D. Дорожня система Ужляндії**

|  |  |
| --- | --- |
| **Input file name:** | roads.in |
| **Output file name:** | roads.out |
| **Time limit:** | 1000 ms |
| **Memory limit:** | 256 M |

Велика і прекрасна країна Ужляндія! Вона складається з ***N*** міст, пронумерованих від 1 до ***N***, і ***M*** доріг між ними. Кожна дорога пов'язує між собою два різних міста та забезпечує пересування між ними. Дорожня система в Ужляндії вельми специфічна. Всі дороги мають двосторонній рух, і між кожною парою міст проходить не більше однієї дороги.

З давніх часів дорожня система Ужляндії задовольняє властивості ***непарності***. З самого початку ця властивість підтримувалося з релігійних міркувань стародавніх ужляндців, а в даний час як данина давній традиції, така ж, як непарна кількість квітів у святковому букеті. Сформулюємо властивість ***непарності***:

Скінчену послідовність номерів міст ***C1,..., CK(K ≥ 2)*** будемо називати ***шляхом***, якщо для будь-якої сусідньої пари елементів послідовність ***Ci, Ci+1 (Ci ≠ Ci+1, для 1 ≤ i < K)*** існує дорога між містами з номерами ***Ci*** ***Ci+1***. Якщо ***C1 = CK***, то такий шлях будемо називати замкнутим. Довжину шляху ***C1, ..., CK*** будемо вважати рівною довжині послідовності, тобто рівній ***K***. Отже, правило ***непарності*** говорить, ***що всі замкнуті шляхи в Ужляндії мають непарну довжину, тобто не існує замкнутого шляху парної довжини***.

Дорожня мережа, зображена на рисунку №1, має властивість непарності, а дорожня система на рисунку №2 не володіє цією властивістю через наявність у ній замкнутого шляху парної довжини 4: 1 2 4 1.

Нещодавно Міністр транспорту Ужляндії вирішив, що існуюча дорожня система неефективна, і необхідно побудувати кілька нових доріг. Причому нова дорожня система, отримана зі старої додаванням деякого числа доріг, повинна мати властивість ***непарності***. Всі нові дороги повинні зв'язувати між собою різні міста. Крім цього, в новій дорожній мережі між кожною парою міст має проходити не більше однієї дороги.

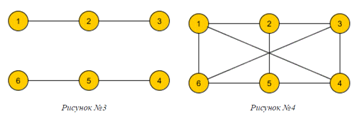
Для поліпшення ефективності дорожньої системи було виділено багато грошей, тому Міністр Ужляндії вирішив побудувати якомога більше нових доріг таким чином, щоб отримана дорожня система задовольняла описаним вище умовам. Але це завдання виявилося досить складним, і міністерство транспорту Ужляндії вирішило звернутися до Вас за консультацією.

Отже, вам дано опис існуючої дорожньої мережі. Вам необхідно знайти максимальне число доріг, яке можна додати до існуючої мережі, не порушуючи вищеописаних властивостей.

**Формат вхідних даних:** Перший рядок вхідного файлу містить цілі числа ***N (1 ≤ N ≤ 10 000)*** і ***M (0 ≤ M ≤ 100000)***. Наступні ***M*** рядків описують дороги Ужляндії. У кожному рядку знаходиться опис рівно однієї дороги. Кожна дорога описується двома цілими числами ***X*** і ***Y (1 ≤ X, Y ≤ N, X ≠ Y)***. Ці числа відповідають номерам міст, зв'язаних дорогою. Міста нумеруються послідовно цілими числами від 1 до ***N***.

Гарантується, що задана дорожня система задовольняє властивості непарності, а також будь-які два міста з'язані не більш ніж однією дорогою .

**Формат вихідних даних:** Вихідний файл повинен містити одне ціле число - максимальну кількість доріг, яку можна додати в існуючу дорожню мережу.

***Пояснення до прикладів:***

Дорожня мережа з першого прикладу приведена на малюнку № 1. Задана дорожня мережа з другого прикладу представлена н​​малюнку №3, а її стан після додавання до неї нових доріг зображено на малюнку №4.

У другому прикладі можна добавити 5 нових доріг: 2 5, 1 4, 1 6, 3 4, 3 6.

***Оцінювання:***

***M = 0*** – не менше 20 балів

***N ≤ 10, M > 0*** – не менше 20 балів

***N ≤ 100, M > 0*** – не менше 40 балів

***N ≤ 1000, M > 0*** – не менше 60 балів

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **roads.in** | **roads.out** |
| 4 4  1 2  2 3  3 4  1 4 | 0 |
| 6 4  1 2  6 5  3 2  4 5 | 5 |

**E. Відео-кафе Ужляндії**

|  |  |
| --- | --- |
| **Input file name:** | video\_cafe.in |
| **Output file name:** | video\_cafe.out |
| **Time limit:** | 1500 ms |
| **Memory limit:** | 128 M |

Як відомо, чемпіонат світу з хокею в 2014 році пройде в Ужляндії. Для успішного проведення заходу приймаючій стороні належить виконати ряд вимог, що пред'являються Міжнародної хокейної федерацією. Хокейні матчі планується провести в м.Ужкачево та м.Ужковель (в Ужляндіїї усі міста починаються на Уж). Дані міста пов'язані між собою прямою автомагістраллю Ужкачево - Ужковель.

Голова Міжнародної хокейної федерації зазначив, що ні на одному з ***N*** кемпінгів, розташованих на автомагістралі Ужкачево - Ужковель немає відео-кафе. Відео-кафе - це елемент придорожнього сервісу, практично нічим не відрізняється від звичайного придорожнього кафе, але в якому створені умови для відео перегляду спортивних подій. Міжнародна хокейна федерація ухвалила, що на ***К*** з ***N*** існуючих кемпінгів необхідно побудувати відео-кафе. Міжнародна хокейна федерація також встановила такі вимоги до розташування кожного відео-кафе:

1. Якщо по дорозі з м.Ужкачево в м.Ужковель зупинитися в деякому відео-кафе, то відстань між даними відео-кафе і ***попереднім*** на автомагістралі відео-кафе (якщо таке є) повинно бути не менше ***А*** км і не більше ***В*** км.

2. Якщо по дорозі з м.Ужкачево в м.Ужковель зупинитися в деякому відео-кафе, то відстань між даними відео-кафе і ***наступним*** на автомагістралі відео-кафе (якщо таке є) повинно бути не менше ***А*** км і не більше ***В*** км.

3. Відстань від м.Ужкачево та м.Ужковель до найближчого відео-кафе не повинно бути менше ***А*** км і не більше ***В*** км.  
   
***Рисунок до прикладу №1: N=5, K=3, A=20, B=70.***

Для кожного ***i***-го побудованого відео-кафе введемо коефіцієнт ***Zi*** - відстань від даного відео-кафе до всіх інших. Міжнародна хокейна федерація встановила, що чим більша сума коефіцієнтів ***Zi***, тим cвоєчасніше мандрівники зможуть отримувати інформацію про проведення хокейних матчів.

Ваше завдання визначити максимальну суму ***Zi***, яка може бути досягнута шляхом зведення ***K*** відео-кафе, що задовольняють всім вимогам Міжнародної хокейної федерації. Відомо що, кожен кемпінг може містити на своїй території не більше одного відео-кафе. Вірне і зворотнє - відео-кафе може розташовуватися тільки на території кемпінгу.

**Формат вхідних даних:** Перший рядок вхідного файлу містить два цілих числа - ***N, K (1 ≤ K ≤ N ≤ 1000)*** відповідно.

Другий рядок вхідного файлу містить два цілих числа ***A, B (1 ≤ A < B ≤ 109)***.  
Третій рядок вхідного файлу містить одне ціле число ***S (1 ≤ S ≤ 109)*** - відстань між м.Ужкачево та м.Ужковель.  
Четвертий рядок вхідного файлу містить ***N*** цілих чисел ***Ci (0 < Ci < S, Ci < Cj якщо i < j)*** - відстань ***i***-го кемпінгу від м.Ужкачево.

**Формат вихідних даних:** Вихідний файл повинен містити одне ціле число - максимальну суму коефіцієнтів ***Zi*** побудови ***K*** відео-кафе. Рішення завжди існує.

***Пояснення до прикладів:***

У першому прикладі кемпінги {1, 2, 4}

У другому прикладі кемпінги {2, 5, 8, 10}.

***Оцінювання:***

***K ≤ 2*** – не менше 15 балів

***NK ≤ 106*** – не менше 30 балів

***N ≤ 50*** – не менше 45 балів

***N ≤ 300*** – не менше 60 балів

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **video\_cafe.in** | **video\_cafe.out** |
| 5 3  20 70  195  30 70 90 135 170 | 420 |
| 10 4  10 28  100  10 19 23 32 47 51 62 74 82 89 | 474 |