Zadanie: KKS
K-krotne składanie

XIII obóz informatyczny, grupa początkująca, dzień 4. Dostępna pamięć: 32 MB. *29.09.2016*

Przemek zajmuje się informatyką i matematyką. Ostatnio zainteresował się tablicami o rozmiarze n, w których występują wszystkie liczby naturalne od 0 do $n-1$ w dowolnej kolejności (i każda z nich występuje dokładnie raz). Takie tablice czasami nazywane są *permutacjami*.

Przemek musi przekształcać permutacje w swoich obliczeniach matematycznych. Bardzo często musi wykonywać przekształcenie nazywane *złożeniem permutacji*. To przekształcenie jest bardzo żmudne i wymaga dużo czasu oraz dobrej pamięci (lub dużo miejsca w brudnopisie). Na szczęście Przemek umie również programować, więc napisał program, który będzie za niego wykonywał przekształcenie permutacji. Oto ten program:

```
#include <cstdio>
int main()
{
    int n, k;
    scanf("%d%d", &n, &k);
    int permutacja[n];
    int tab[k][n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &permutacja[i]);
        tab[0][i] = permutacja[i];
    }
    for (int i = 1; i < k; i++) {
        for (int j = 0; j < n; j++) {
            tab[i][j] = tab[i - 1][permutacja[j]];
        }
    }
    for (int i = 0; i < n; i++) {
        printf("%d ", tab[k - 1][i]);
    }
    return 0;
}
```

Niestety po skompilowaniu oraz uruchomieniu okazało się, że program wymaga zbyt dużo pamięci, a czas działania jest zbyt długi. To spowodowało, że Przemek mógł wykonywać przekształcenia jedynie na bardzo małych permutacjach. Możesz się o tym przekonać sam na własnej skórze. Po wysłaniu powyższego algorytmu na sprawdzarkę, dostaniesz jedynie 10 punktów.

Przemek był bardzo smutny, ale przypomniał sobie o swoim koledze, który jest bardziej doświadczonym programistą. Jesteś nim Ty i Twoim zadaniem jest napisanie własnego algorytmu (lub przerobienie powyższego) tak, aby działał on wystarczająco szybko oraz mieścił się w podanym limicie pamięci dla ograniczeń podanych w sekcji *Wejście*. Jeśli jednak nie potrafisz tego zrobić to możesz wysłać powyższy algorytm na sprawdzarkę.

Jeśli uda Ci się zoptymalizować jedynie pamięć, a czas działania pozostanie taki sam to dostaniesz 50 punktów.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n oraz k ($1 \leq k \leq n \leq 10^6$), oznaczające odpowiednio rozmiar permutacji oraz parametr k oznaczający liczbę iteracji pierwszej pętli w algorytmie Przemka. Drugi wiersz przedstawia tablicę, która jest *permutacją* o rozmiarze n. Znajduje się w nim n parami różnych liczb naturalnych z przedziału $[0, n-1]$.

Wyjście

Pierwszy i jedyny wiersz standardowego wyjścia powinien zawierać n różnych liczb naturalnych z przedziału $[1, n]$. Ten wynik powinien być taki sam jak wynik algorytmu napisanego przez Przemka.

Przykład

Dla danych wejściowych:

```
4 2
1 2 3 0
```

poprawnym wynikiem jest:

```
2 3 0 1
```