Problem H
Parking

III OTWARTE MISTRZOSTWA OPOLA W PROGRAMOWANIU ZESPOŁOWYM

Na zakładowym parkingu wszyscy pracownicy parkują w czasie godzin pracy swoje samochody. W zakładzie pracuje *K* osób. Kolejne miejsca parkingowe są umieszczone jedno obok drugiego i każdemu można przypisać liczbę określającą numer pracownika, którego auto stoi na tym miejscu (1…*K*). Istnieje także zasada, że samochód dyrektora i wicedyrektora nie mogą stać koło siebie. Napisać program podający wszystkie możliwe rozmieszczenia *K* samochodów tak, aby koło siebie nie znajdowały się dwa wymienione samochody.

Zadanie

Napisz program, który dla podanych danych wejściowych wypisze wszystkie możliwe sposoby zaparkowania samochodów na parkingu, z uwzględnieniem założenia, iż dwa podane samochody nie mogą stać koło siebie.

Wejście

Program będzie sprawdzany przy pomocy zestawu *N* testów. Pierwszy wiersz standardowego wejścia zawiera liczbę testów. Od drugiego wiersza umieszczono dane wejściowe dla kolejnych testów. Dane wejściowe dla jednego testu to rozdzielone spacjami trzy liczby naturalne zawarte w jednym wierszu pobieranym ze standardowego wejścia. Pierwsza liczba określa ilość zaparkowanych samochodów, a druga i trzecia numery samochodów, które nie mogą parkować koło siebie. Ilość testów nie może przekraczać 10, a ilość samochodów w każdym teście mieści się w zakresie od 3 do 8.

Wyjście

Na standardowym wyjściu, dla każdego testu, program powinien wyprowadzić w kolejnych liniach ciągi liczb naturalnych reprezentujące kolejne możliwości ustawienia samochodów. Znakiem rozdzielającym liczby umieszczone w jednej linii jest spacja. Kolejne testy powinny być rozdzielone pustą linią. Kolejne sposoby rozmieszczenia samochodów powinny być wyprowadzane na wyjście w porządku leksykograficznym (słownikowym).

Przykład

Dla 2 testów:

test 1: na parkingu znajdują się cztery samochody, drugi i trzeci nie mogą stać koło siebie
test 2: na parkingu znajduje się pięć samochodów, drugi i piąty nie mogą stać koło siebie

dane wejściowe są podane następująco:

```
2
4 2 3
5 2 5
```

Program powinien zwrócić w wyniku:

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

2 4 3 1
3 1 2 4
3 1 4 2
3 4 1 2
3 4 2 1
4 2 1 3
4 3 1 2

1 2 3 4 5
1 2 3 5 4
1 2 4 3 5
1 2 4 5 3
1 3 2 4 5
1 3 5 4 2
1 4 2 3 5
1 4 5 3 2
1 5 3 2 4
1 5 3 4 2
1 5 4 2 3
1 5 4 3 2
2 1 3 4 5
2 1 3 5 4
2 1 4 3 5
2 1 4 5 3
2 1 5 3 4
2 1 5 4 3
2 3 1 4 5
2 3 1 5 4
2 3 4 1 5
2 3 4 5 1
2 3 5 1 4
2 3 5 4 1
2 4 1 3 5
2 4 1 5 3
2 4 3 1 5
2 4 3 5 1
2 4 5 1 3
2 4 5 3 1
3 1 2 4 5
3 1 5 4 2
3 2 1 4 5
3 2 1 5 4
3 2 4 1 5
3 2 4 5 1
3 4 2 1 5
3 4 5 1 2
3 5 1 2 4
3 5 1 4 2
3 5 4 1 2
3 5 4 2 1
4 1 2 3 5
4 1 5 3 2
4 2 1 3 5
4 2 1 5 3
4 2 3 1 5
4 2 3 5 1
4 3 2 1 5
4 3 5 1 2
4 5 1 2 3
4 5 1 3 2
4 5 3 1 2
4 5 3 2 1
5 1 2 3 4
5 1 2 4 3
5 1 3 2 4
5 1 3 4 2
5 1 4 2 3
5 1 4 3 2
5 3 1 2 4
5 3 1 4 2
5 3 2 1 4
5 3 2 4 1
5 3 4 1 2
5 3 4 2 1
5 4 1 2 3
5 4 1 3 2
5 4 2 1 3
5 4 2 3 1
5 4 3 1 2
5 4 3 2 1