1.Szyfr Cezara

Szyfr Cezara jest to szyfr za pomocą, którego Juliusz Cezar szyfrował swoje listy do Cycerona. Jako ciekawostkę można podać, że szyfr ten był podobno używany jeszcze w 1915 roku w armii rosyjskiej, gdyż tylko tak prosty szyfr wydawał się zrozumiały dla sztabowców.

Każdą literę tekstu jawnego zamieniamy na literę przesuniętą o 3 miejsca w prawo. I tak literę A szyfrujemy jako literę D, literę B jako E itd. W przypadku litery Z wybieramy literę C. W celu odszyfrowania tekstu powtarzamy operację tym razem przesuwając litery o 3 pozycje w lewo.

Input

Na wejściu podajemy łańcuch dużych liter bez polskich znaków.

Output

Na wyjściu otrzymujemy zaszyfrowany tekst używając Szyfru Cezara.

Example

```
Input:
ABC DEF
TERA EST ROTUNDA

Output:
DEF GHI
WHUD HVW URWXQGD
```

2.Replikacja łańcucha

Napisz funkcję:

```
char* replicate(char *S, int n);
```

która utworzy nowy łańcuch składający się z `n` kopii łańcucha `S` i umieści w nowej dynamicznie alokowanej tablicy znaków, do której zwróci wskaźnik.

Input

W pierwszej linii liczba testów *t*, w kolejnych liniach liczba `n>0` i łańcuch znaków składający się z małych i wielkich liter.

Output

W kolejnych `t` wierszach powielone łańcuchy.

Example

Input:
```
2
4 aB
1 x
```

Output:
```
aBaBaBaB
x
```

3.Concat

Napisz funkcję:

```
char* my_strcat(char *, char *);
```

która sklei ze sobą dwa łańcuchy i umieści w nowej dynamicznie alokowanej tablicy znaków, do której zwróci wskaźnik.

Input

W pierwszej linii liczba testów t, w kolejnych liniach po dwa łańcuchy znaków odzielone spacją.

Output

W każdej linii jeden łańcuch, wynik działania funkcji my_strcat

Example

Input:

```
4
a b
abs sfd
ewr w
wqeqweqweq eqweqwe
```

Output:

```
ab
abssfd
ewrw
wqeqweqweqeqweqwe
```

4.String compare

Mamy alfabet `A` złożony z trzech liter: `c, a, b`. Napisz funkcję

```
int my_strcmp(char S1[], char S2[]);
```

która porówna ze sobą łańcuchy `S1` i `S2` zapisane w alfabecie `A`. Wynikiem powinno być:

```
 0, jeśli S1=S2;
-1, jeśli S1 jest leksykograficznie mniejszy niż S2;
 1, jeśli S1 jest leksykograficznie większy niż S2.
```

Porządek leksykograficzny jest ustalany zgodnie z alfabetem `A`. Niech `n` będzie numerem pierwszego znaku na którym różnią się `S1` i `S2`. `S1` jest leksykograficznie mniejsze niż `S2`, jeśli `n`-ty znak w `S1` występuje w `A` wcześniej niż `n`-ty znak w `S2` lub słowo `S1` ma tylko `n-1` znaków.

Input

W pierwszej linii liczba testów t, w kolejnych liniach po dwa łańcuchy znaków odzielone spacją. Maksymalna długość łańcucha wynosi 999 znaków.

Output

W każdej linii jedna liczba, wynik działania funkcji my_strcmp.

Example

Input:

```
5
aa cc
ac a
c  ac
bc ab
bc ac
```

Output:

```
1
1
-1
1
1
```

5.Glue

Napisz funkcję

```
int* glue(int **T1, int l, int k)
```

która sklei wszystkie elementy tablicy `T1`, zawierającej `l` wskaźników do tablic `k` liczbowych w jedną tablicę `lk` liczbową. Funkcja zwraca wskaźnik do utworzonej tablicy. Tablica będąca parametrem funkcji powinna być niszczona.

Input

W pierwszej linii liczba testów `t`. W każdym teście najpierw dwie liczby `l` i `k`. Następnie w każdym z `l` wierszy znajduje się `k` liczb.

Output

Dla każdego testu w pierwszym wierszu liczba `l k` i następnie w kolejnym wierszu `l k` liczb.

Example

```
Input:
3
1 1
1
2 2
1 2
3 4
3 1
1
2
2


Output:
1
1 2 3 4
1 2 2
```