Zadanie 1 (10 pkt. na pracowni, później 5 pkt.). Rozważamy napisy składające się z liter alfabetu łacińskiego (małych i wielkich), cyfr oraz znaków spacji i kropki. Twoim zadaniem będzie napisać funkcje, które będą kodowały oraz dekodowały takie napisy w opisany poniżej sposób.

Przypiszmy kolejnym znakom liczby z przedziału od 0 do 63 następująco:

- a → 0, b → 1, ..., z → 25;
- A → 26, B → 27, ..., Z → 51;
- 0 → 52, 1 → 53, ..., 9 → 61;
- spacja → 62, . → 63.

Każda z powyższych liczb mieści się na 6 bitach, a więc w jednej liczbie typu unsigned int jesteśmy w stanie pomieścić 5 takich liczb. Kodowanie będzie polegało na dobieraniu po 5 kolejnych znaków oraz zamianie ich na jedną liczbę typu unsigned int.

Przykładowo, napis "Ala ma 3 koty." może zostać zakodowany jako ciąg liczb

```
217580250 184254336 16614606
```

Zapis binarny pierwszej z nich to

```
001100 111110 000000 001011 011010
```

co kolejno (od najmniej znaczących pozycji) oznacza wartości liczb odpowiadających znakom A, l, a, spacji oraz m.

Napisz funkcje o sygnaturach:

- void encode(const char *from, unsigned int *to),
- void decode(int n, const unsigned int *from, char *to),

które odpowiednio zakodują i odkodują treść znajdującą się we from (czytając odpowiednio do bajtu zerowego lub kolejnych n komórek), rezultat zapisując do to. Wywołując funkcje należy oczywiście zadbać o odpowiednią długość buforów przekazywanych jako to, w przypadku decode uwzględniając też bajt zerowy, który powinna ona zapisywać.

Za napisanie jednej z powyższych funkcji można otrzymać połowę punktów.

Przydatne mogą też być funkcje o sygnaturach:

- unsigned int charToInt(char c),
- char intToChar(unsigned int nr),

kodujące i dekodujące pojedynczy znak (w drugim przypadku – z najmniej znaczących bitów argumentu)... albo inne, podobne.

W funkcji main przetestuj działanie zaimplementowanych funkcji. Zauważ, że w przypadku napisów długości niepodzielnej przez 5 dekodowanie nie jest funkcją odwrotną do kodowania, bo "dopisuje" na końcu pewną liczbę liter a – nie przejmuj się tym.

Zadanie 2 (10 pkt.). Dany jest zbiór $n \leq 20$ piłeczek, z których każda jest pokolorowana na jeden lub kilka kolorów (których łącznie jest 60). Twoim zadaniem jest napisać program, który policzy, ile maksymalnie piłeczek możemy wybrać ze zbioru tak, aby żadne dwie z nich nie miały ani jednego wspólnego koloru.

Dane wejściowe składają się z liczby piłeczek n, po której następuje n liczb a_1, a_2, ..., a_n.
Liczba a_i oznacza **maskę bitową** kolorów i-tej piłeczki, gdzie j-ty bit jest równy 1 dokładnie wtedy, gdy i-ta piłeczka jest pokolorowana na kolor o numerze j.

Przykładowo, liczba 25 ma zapis bitowy 11001, co oznacza, że opisana nią piłeczka jest pomalowana na kolory

o numerach 0, 3 i 4 (bity numerujemy od 0, od najmniej znaczącego).

Zadanie rozwiąż rozważając wszystkie możliwe podzbiory piłeczek, a następnie dla każdego z nich sprawdzając, czy żaden kolor nie powtarza się na piłeczkach z tego podzbioru. Do rozważenia podzbiorów również użyj masek bitowych, tj. przejrzyj wszystkie liczby od 0 do 2^*n*-1, traktując ich zapisy bitowe jako tablice charakterystyczne kolejnych podzbiorów (jest to reprezentacja zbiorów jak w programie o sicie Eratostenesa na wykładzie, tylko na pojedynczej liczbie, a nie całej ich tablicy).

Przykładowo, dla danych:

```
4
3 12 6 16
```

odpowiedzią jest 3, gdyż kolory na piłeczkach opisanych liczbami 3, 12 i 16 nie pokrywają się. Za to dla danych:

```
5
14 13 11 7 15
```

odpowiedzią jest 1, gdyż wszystkie pary piłeczek mają jakiś wspólny kolor.

Zadanie 3 (10 pkt.).

W tym zadaniu należy policzyć wybrane wyrazy ciągu wartości ze zbioru {0, 1, 2, 3} zadanego w następujący sposób:

- Pierwszych 16 wyrazów (o indeksach od 0 do 15) jest danych, kolejne zadane są wzorem $C_p = (C_q \times C_r + C_s) \bmod 4$ dla pewnych indeksów q, r, s.
- Do określenia tych indeksów potrzebna jest ustalona permutacja σ 16 elementów (również numerowanych od 0 do 15), która najpierw określa pomocniczy ciąg $Z_i = σ^{(i)}(0)$ (tu "potęga" oznacza złożenie permutacji), tj. $Z_0 = 0$ i $Z_{(i+1)} = σ(Z_i)$ dla $i \geq 0$.
- Dla indeksów p od 16 do 255, indeksy q, r, s to kolejne wartości Z_i w ten sposób, że (pomijamy operację "mod 4" dla przejrzystości) $C_{16} = C_{(Z_0)} \times C_{(Z_1)} + C_{(Z_2)}$, $C_{17} = C_{(Z_3)} \times C_{(Z_4)} + C_{(Z_5)}$ itd., tj. aby określić kolejny wyraz ciągu C, bierzemy jako indeksy we wzorze rekurencyjnym kolejne trzy (po ostatnio wykorzystanym) wyrazy ciągu Z.
- Dla indeksów p od 256 do 16^3-1, indeksy q, r, s mają postać $Z_j \times 16 + Z_{(j+1)}$. Takie wyrażenie wykorzystujące dwa kolejne wyrazy ciągu Z daje indeks q, dwa następne – r, dwa jeszcze następne – s; razem by określić kolejny wyraz C "zużywamy" tu 6 wyrazów Z.
- Ogólniej, dla p od 16^k do $16^{(k+1)}-1$, indeksy q, r, s dane są wyrażeniami $Z_j \times 16^{(k-1)} + Z_{(j+1)} \times 16^{(k-2)} + ... + Z_{(j+k-2)} \times 16 + Z_{(j+k-1)}$, a dla określenia kolejnego wyrazu C potrzeba $3k$ wyrazów Z.

Wejście

Na wejściu znajdują się trzy wiersze po 16 nieujemnych liczb każdy. W pierwszym wierszu znajdują się indeksy p_i wyrazów ciągu, które należy policzyć; w drugim – początkowe wyrazy C_0, ..., C_{15} ciągu (ze zbioru {0, 1, 2, 3}. Trzeci wiersz to opis permutacji σ **jako cyklu**, tj. np. jeśli zaczyna się on od liczb "7 3", to nie oznacza to, że σ(0) = 7 i σ(1) = 3, tylko że σ(7) = 3, czyli 7 "przechodzi na" 3 (dodatkowo ostatnia liczba w wierszu "przechodzi na" tę na początku). Nie ma gwarancji, że ten wiersz zaczyna się od 0, natomiast na pewno opisuje on poprawną permutację cykliczną, tj. każda z liczb od 0 do 15 pojawia się w nim dokładnie raz.

Wyjście

Na wyjściu powinien znaleźć się jeden wiersz zawierający kolejne wartości żądanych $C_{(p_i)}$. Na końcu wiersza mogą znajdować się nadmiarowe spacje.

Uwaga! W tym zadaniu jest limit pamięciowy ~~8 MB~~ 11 MB, na pewno da się przydzielić tablicę o rozmiarze 8 MB. p_i nie przekraczają 30×10^6, a w pierwszych czterech testach – 8×10^6.

Przykłady

Przykład A

Wejście

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 2 3 3 1 0 3 1 1 2 1 0 1 0 0 0
14 10 6 4 5 11 15 12 7 0 3 2 9 13 8 1
```

Wyjście

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

Komentarz

Tu należy przepisać początkowe wyrazy, a permutacja nie ma znaczenia.

Przykład B

Wejście

21 34 57 83 84 122 128 135 142 158 191 192 197 207 243 254
0 1 3 2 2 0 3 3 1 2 1 0 0 0 0 0
5 3 11 15 12 0 14 1 10 8 4 7 2 9 6 13

Wyjście

0 3 0 0 0 0 1 1 2 2 0 1 0 0 0 2

Komentarz

Tu *p_i* są pomiędzy 16 a 255...

Przykład C

Wejście

391 639 1076 1155 1240 1291 1638 1799 2299 2352 2765 2795 2902 3204 3743 3888
0 1 1 1 1 0 0 2 3 1 1 2 0 2 2 1
4 11 0 8 12 10 3 6 1 2 14 7 15 9 13 5

Wyjście

3 3 1 1 3 1 0 3 1 3 1 1 0 1 3 3

Komentarz

... a tu – pomiędzy 256 a 4095.