# Zadanie: KOL
# Koledzy Bajtazara

Olimpiada Informatyczna

**ONTAK 2014, dzień drugi. Plik źródłowy `kol.*` Dostępna pamięć: 128 MB.** *6.8.2014*

Bajtazar mieszka w ogromnym domu, w którym znajduje się bardzo wiele pokoi*. Bajtazar chciałby urządzić imprezę, ale będąc leniwym z natury, nie ma ochoty zastanawiać się, kogo zaprosić (a znajomych ma bardzo wielu) ani też ile będzie potrzebował pokoi. Te szczegóły pozostawił Tobie.

Znajomi Bajtazara będą zjawiać się kolejno przez cały wieczór. Niestety, nie wszyscy z nich lubią się między sobą. Każdy z przychodzących gości:

- dostaje kolejny numer będący liczbą naturalną (pierwszy gość dostaje numer 0),
- mówi Bajtazarowi, kogo spośród *aktualnie* obecnych na imprezie nie lubi,
- jest przez Bajtazara kierowany do jednego z pokoi (pokoje również są numerowane od 0), przy czym dwie nielubiące się osoby są zawsze kierowane do różnych pokoi.

Założyłeś się z Bajtazarem, że przyślesz mu takich kolegów, aby użył on jak najwięcej pokoi (wybór masz praktycznie nieograniczony, Bajtocja to duży kraj...). Twoim zadaniem jest udowodnić mu, że tych samych kolegów mógłby upakować w znacznie mniejszej liczbie pokoi, gdyby tylko na samym początku znał listę gości, wiedział, którzy z nich się nie lubią i starannie zaplanował rozmieszczenie gości w pokojach.

## Komunikacja programu

Twój program nie powinien nic czytać z wejścia ani wypisywać na wyjście. Zamiast tego zostanie on skompilowany z biblioteką oceniającą, która będzie symulowała zachowanie Bajtazara – przydzielanie przychodzących kolegów do pokoi. Na początku swojego programu umieść dyrektywę:

```
#include "kol.h"
```

Biblioteka udostępnia trzy funkcje:

- `void init()` — funkcję tę Twój program powinien wykonać (koniecznie!) raz, na początku programu.
- `int create(vector<int> wrogowie)` — wywołanie tej funkcji oznacza wysłanie kolegi na imprezę. Kolega wysłany w $k$-tym wywołaniu otrzymuje numer $k-1$. Wektor `wrogowie` powinien zawierać numery kolegów (mniejsze od numeru przychodzącego), którzy nie mogą trafić do tego samego pokoju. Funkcja zwraca numer pokoju, do którego Bajtazar przydzielił kolegę. Numery pokojów użytych przez Bajtazara będą zawsze tworzyć pewien spójny przedział $[0, x]$.
- `void answer(vector<int> pokoje)` — ta funkcja powinna zostać wywołana raz, na końcu działania programu. Oznacza ona, że zakończyłeś przysyłanie kolegów i pokazujesz Bajtazarowi swój przydział do pokoi, zapisany w wektorze `pokoje`. Dokładniej, liczba `pokoje[i]` powinna oznaczać numer pokoju, do którego według Ciebie powinien trafić kolega numer $i$.

Na imprezę możesz przysłać nie więcej niż 5000 kolegów. Łączna liczba antagonizmów (czyli łączna długość wektorów `wrogowie`) nie może przekroczyć 300 000.

## Ocenianie

W tym zadaniu różne testy odpowiadają różnym strategiom przydziału pokoi stosowanym przez Bajtazara. Aby otrzymać punkty za test, Twoja odpowiedź przede wszystkim musi być poprawna (dwaj nielubiący się koledzy nie mogą trafić do jednego pokoju). Liczba różnych pokoi, których użył Bajtazar, porównywana jest z liczbą w Twoim rozwiązaniu. Punktacja testu zależy od różnicy między tymi liczbami. Otrzymasz:

- 20% punktów za test, jeśli Bajtazar użyje przynajmniej o 1 pokój więcej niż Ty,
- 30% punktów, jeśli o 2 pokoje więcej,
- ...
- 100% punktów, jeśli liczba pokoi Bajtazara przewyższy Twoją o co najmniej 9.

Jedną z możliwych strategii Bajtazara jest wysyłanie każdego gościa do pokoju o najmniejszym możliwym numerze. Strategia ta będzie używana w teście wartym 30% punktów. Pozostałych strategii nie ujawniamy – spróbuj napisać program, który oszuka dowolnie postępującego Bajtazara.

*Jeśli wierzyć legendzie, dom budował jego pradziadek, Hilbert.

## Przykładowa biblioteka

W dziale Pliki na SIO znaleźć można przykładową bibliotekę `kolzaw.cpp`, która implementuje podane trzy funkcje. W tej przykładowej wersji Bajtazar zachowuje się wyjątkowo niemądrze, każdego z kolegów przydzielając do nowego pokoju. Funkcja `answer()` sprawdza, ilu pokoi użył Twój program i wypisuje porównanie na standardowe wyjście.

Aby przetestować swój program na przykładowej strategii Bajtazara, skompiluj go komendą podobną do poniższej (zakładamy, że Twoje rozwiązanie znajduje się w pliku `kol.cpp`:

```
g++ -O2 -static kol.cpp kolzaw.cpp -o kol
```

Spowoduje to utworzenie pliku wykonywalnego o nazwie `kol`.