Zadanie 1 (10 pkt. na pracowni, później 5 pkt.). Liczby Catalana C_n opisują m.in. liczbę poprawnych rozstawień n par nawiasów, i są opisane następującą relacją rekurencyjną: $C_0 = 1$, a dla $n > 0$ zachodzi $C_n = C_0 C_{n-1} + C_1 C_{n-2} + \ldots + C_{n-1} C_0$.

Inna relacja rekurencyjna, którą spełniają liczby Catalana, to $C_n = 2(2n-1)/(n+1) \times C_{n-1}$ dla $n > 0$.

Napisz trzy funkcje, które będą obliczać C_n różnymi metodami:

- wprost z pierwszej relacji (rekurencyjnie),
- z pierwszej relacji ze spamiętywaniem (tj. iteracyjnie wpisując C_0, C_1 itd. w kolejne komórki tablicy),
- z drugiej relacji (rekurencyjnie, choć tu już nie ma to tak dużego znaczenia).

Program powinien przyjmować jeden lub dwa argumenty wywołania. Jeśli są dwa, to jeden z nich (dowolny!) musi być napisem -n, -m lub -f (warto użyć strcmp) i będzie decydował, która z trzech kolejnych ww. metod zostanie wykorzystana do obliczenia C_n, gdzie nieujemne n należy odczytać (np. przy użyciu atoi) z pozostałego argumentu wywołania. Jeśli argument jest tylko jeden, to jest to oczywiście n, a obliczenie należy wykonać metodą ze spamiętywaniem (tj. jak gdyby obecny był argument -m). Wywołania z niepoprawnymi argumentami (w tym gdy jest ich za dużo, za mało, n jest ujemne itp.) powinny skutkować wyłącznie wypisaniem odpowiedniego komunikatu.

Efektem poprawnego wywołania programu ma być wypisanie C_n na standardowe wyjście. Wartości liczb Catalana obliczaj i zapisuj w typie long long – dzięki temu możliwe będzie otrzymanie poprawnych wyników nawet dla $n = 33$. Porównując (dla różnych wartości liczbowych w miejsce X) efekty wywołań

```
time ./a.out -n X
time ./a.out -m X
time ./a.out -f X
```

znajdź taką wartość argumentu, dla której najwolniejsza metoda będzie wyraźnie "odstawać" od pozostałych, ale jeszcze będzie się dało doczekać końca jej obliczeń. (Obliczenia trwające zbyt długo możesz przerywać kombinacją Ctrl-C.)

Zadanie 2 (10 pkt.). Podział skończonego zbioru to rodzina jego niepustych, parami rozłącznych podzbiorów, których suma daje cały oryginalny zbiór. Przykładowo, zbiór {1, 2, 3, 4} ma 15 różnych podziałów, a trzy z nich to:

- {1, 2, 3, 4};
- {1}, {2}, {3}, {4};
- {1, 2, 4}, {3}.

Napisz program, który będzie wymagał dokładnie jednego argumentu wywołania, który (np. przy użyciu atoi) będzie się dało zinterpretować jako liczbę całkowitą dodatnią n (wszystkie inne wywołania – z niewłaściwą liczbą argumentów albo z niepoprawnym argumentem, w tym wyglądającym jak liczba ujemna), i wydrukuje wszystkie podziały zbioru $\{1, 2, \ldots, n\}$.

Wykorzystaj następujące obserwacje (nie musisz ich uzasadniać):

- zbiór {1} ma tylko jeden podział,
- mając jakiś podział zbioru $\{1, 2, \ldots, m-1\}$, możemy przerobić go na podział zbioru $\{1, 2, \ldots, m\}$ albo dokładając m do dowolnego z podzbiorów w tym podziale, albo dokładając nowy podzbiór $\{m\}$; przechodząc tak po wszystkich podziałach zbioru "mniejszego" otrzymamy każdy z podziałów zbioru "większego" dokładnie na jeden sposób.

Dany podział można zareprezentować tabelą p taką, że $p[i]$ to numer podzbioru, do którego należy $i+1$. Przykładowo ostatni ww. podział zbioru czteroelementowego reprezentuje taki stan p, że $p[0] = p[1] = p[3] \neq p[2]$ (konkretne wartości są arbitralne).

Każdy podział powinien zostać wydrukowany w osobnym wierszu, z pionowymi kreskami oddzielającymi

podzbiory i spacjami oddzielającymi elementy. Przykładowo, wywołanie ./a.out 3 może mieć taki efekt:

```
1 2 3
1 2 | 3
1 3 | 2
1 | 2 3
1 | 2 | 3
```

ale kolejność (podziałów, podzbiorów w podziale, elementów w podzbiorze) może być dowolna. Spróbuj nie korzystać ze zmiennych nielokalnych, zamiast tego przekazuj tablicę jako argument, zwracaj liczbę wygenerowanych podzbiorów itp.

Wersja uproszczona (7 pkt.): jeśli nie chcesz męczyć się z drukowaniem wyjścia, niech program po prostu policzy podziały i wydrukuje ich liczbę. (Są to liczby Bella. Dodatkowo $B_0 = 1$, ale nie potrzebujemy się dziś zastanawiać, czy zbiór pusty w ogóle może mieć jakieś podziały i dlaczego pusta rodzina podzbiorów – ale nie rodzina zawierająca zbiór pusty! – to właśnie taki podział.)

Zadanie 3 (10 pkt.).

Dane jest drzewo genealogiczne zawierające potomków pana Korzeniowskiego. Opis potomstwa pana Korzeniowskiego podany jest w następujący sposób:

- najpierw występuje liczba a_K – liczba dzieci pana Korzeniowskiego,
- następnie występuje a_K opisów potomstwa dzieci pana Korzeniowskiego, w kolejności od najstarszego. Opisy te są w tym samym formacie co opis potomstwa pana Korzeniowskiego.

Jeśli któryś z potomków jest bezdzietny, jego opis stanowi pojedyncza liczba 0.

Na przykład, jeśli pan Korzeniowski ma trójkę dzieci – z czego najstarsze ma dwójkę, średnie nie ma, a najmłodsze ma piątkę dzieci – i nie ma prawnuków, to opisem potomstwa jest 3 2 0 0 0 5 0 0 0 0 0.

Określamy też stopień bycia potomkiem za pomocą liczb naturalnych. Bycie potomkiem stopnia 0 osoby *O* jest tożsame z byciem osobą *O*. Potomek stopnia 1 jest dzieckiem, potomek stopnia 2 – wnukiem, stopnia 3 – prawnukiem i tak dalej. Dla powyższego przykładu pan Korzeniowski ma *1* potomka stopnia 0 (siebie), *7* potomków stopnia 2 i *3* potomków stopnia 1, jego najstarsze dziecko na dwóch potomków stopnia 1, a najmłodsze – *5* potomków stopnia 1.

Twoim zadaniem jest wczytać liczbę *n* oraz podane drzewo genealogiczne i policzyć osoby, dla których prawdą jest, że:

- każdy bezdzietny potomek takiej osoby jest jej potomkiem stopnia **co najmniej n**,
- ma bezdzietnego potomka stopnia *n*.

Np. dla *n* = 2 oznacza to pytanie liczbę osób, które mają bezdzietne wnuki, ale ich wszystkie dzieci mają dzieci.

Wejście

Wejście składa się z dwóch wierszy. W pierwszym wierszu znajduje się liczba 0 ≤ *n* ≤ 10^6. W drugim wierszu jest opis drzewa genealogicznego. We wszystkich testach jest nie więcej niż 10^6 członków rodziny. Załóż, że każda osoba w tym drzewie ma dokładnie jednego rodzica z tego drzewa.

Wyjście

Wyjście składa się z pojedynczej liczby naturalnej – odpowiedzi na pytanie.

Przykłady

Przykład A

Wejście

```
2
3 3 0 0 0 5 0 0 0 0 0 2 0 0
```

Wyjście

```
1
```

Komentarz

Pytamy o liczbę osób posiadających bezdzietne wnuki, których wszystkie dzieci mają dzieci. Pan Korzeniowski ma trójkę dzieci, każde z nich ma dzieci i żadne z nich nie ma wnuków, wobec czego wyłącznie pan Korzeniowski jest liczony do wyniku.

Przykład B

Wejście

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

Wyjście

6

Przykład C

Wejście

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

Wyjście

3

Wskazówka

W implementacjach rekurencyjnych szczególnie wygodne może być wykorzystanie zmiennych nielokalnych, tj. zadeklarowanych poza jakimkolwiek blokiem kodu i "widocznych" z każdej funkcji (o ile deklaracja takiej zmiennej jest w kodzie przed definicją funkcji). Używanie takich zmiennych to często nie jest dobry pomysł, ale o tym (i innych powiązanych kwestiach) jeszcze powiemy na wykładzie w swoim czasie. Co ważne, tablice tak deklarowane muszą mieć konkretną długość, tj. niezależną od żadnych zmiennych (czyli nie mogą to być VLA).