Zadanie 1 (10 pkt. na pracowni, później 5 pkt.). Napisz program, który rekurencyjnie oblicza *n*-ty wyraz ciągu zadanego dla $n \geq k$ równością $f(n) = a\_0 * f(n\text{-}1) + a\_1 * f(n\text{-}2) + ... a\_(k\text{-}1) * f(n\text{-}k)$. Program powienien przyjmować $2k+1$ argumentów:

- pierwszy argument to liczba $k > 0$;
- kolejne k argumentów to liczby $a\_0, a\_1, ..., a\_(k\text{-}1)$;
- ostatnie k argumentów to liczby $f(0), f(1), ..., f(k\text{-}1)$.

Poza pierwszym, argumenty mogą być niecałkowite (przy typowej konfiguracji powłoki pewnie będzie należało użyć kropki jako separatora części ułamkowej), do ich zinterpretowania możesz użyć *atof* z *stdlib.h*. Nieujemną liczbę *n* program powinien wczytywać ze standardowego wejścia.

Zadbaj, aby twój program sprawdzał poprawność podanych argumentów.

Zadanie 2 (10 pkt.). Dla ustalonego różnowartościowego ciągu $S = (x\_0, x\_1, ..., x\_(n\text{-}1))$ definiujemy porządek pomiędzy jego niepustymi podciągami *A, B* w następujący sposób: $A < B$ wtedy i tylko wtedy gdy

- *A* jest prefiksem *B*, lub
- *a* jest wcześniej niż *b* w *S*, gdzie *a, b* to znaki w *A, B* na najwcześniejszej pozycji, na której się one różnią.

Przykładowo, dla ciągu S = (a, b, f, c) zachodzi (a, b, c) > (a, b, f, c) oraz (b) < (f) < (f, c).

Napisz program, który dla danego ciągu znaków oblicza jego podciągi z uwzględnieniem powyższej kolejności. Pierwszym, obowiązkowym argumentem wywołania programu jest napis stanowiący ciąg *S* – możesz założyć, że będzie on dość krótki (do ok. 20 znaków), żeby wykonanie programu nie było zbyt długie. Ewentualne dalsze argumenty określają szczegóły działania programu.

Wywołanie z flagą *-a*, np. *./a.out abcd -a* powinno wypisać wszystkie niepuste podciągi w ww. porządku, czyli w tym przypadku

```
a
ab
abc
abcd
abd
ac
acd
ad
b
bc
bcd
bd
c
```

```
cd
d
```

Wywołanie z flagą *-i [indeks]*, np. *./a.out adc -i 2* powinno wypisać ten podciąg, który ww. porządku jest na pozycji *[indeks]* (liczonej od 1), a więc w tym przypadku *ad*.

Wreszcie wywołanie bez dodatkowej flagi powinno wypisać losową liczbę z odpowiedniego zakresu i, w kolejnym wierszu, podciąg z tej pozycji. Do losowania użyj funkcji *rand()* z nagłówka *stdlib.h* (a żeby faktycznie zobaczyć zmienność losowania, najpierw wywołaj *srand(time(NULL))*, dołączając jeszcze *time.h* – o szczegóły tego, co tu się dzieje, możesz zapytać osobę prowadzącą pracownię).

Wszystkie niepoprawne wywołania (bez argumentów, z dodatkowym argumentem po *-a*, bez dodatkowego argumentu po *-i*, z argumentem po *-i* nie dającym się zinterpretować jako liczba dodatnia etc.) powinny skutkować wypisaniem odpowiedniego komunikatu o błędzie i wyjściem.

Wskazówka: warto skorzystać z rekurencji i następującej obserwacji: wszystkie podciągi zawierające *x_0* będą w porządku wcześniej, niż pozostałe; w dodatku będą one zaczynać się od *x_0*, po którym będą następować kolejne podciągi (*x_1*, *x_2*, ..., *x_(n-1)*).

(Kiedy *S* jest uporządkowane alfabetycznie, jak w jednym z powyższych przykładów, to nieprzypadkowo otrzymujemy porządek leksykograficzny. Pusty podciąg wstawiony na początek porządku dalej będzie do niego pasował, ale pomijając go, łatwiej było napisać wskazówkę... Jeśli chcesz, możesz spróbować samemu wymyślić rekurencję, która wygeneruje również pusty podciąg – wtedy, dla spójności, będziemy zakładać, że ma on indeks 0.)

Uwaga: Konieczne może się okazać przekazywanie tablic jako argumentów funkcji. Możesz zapytać o szczegóły osobę prowadzącą pracownię, poczekać do wykładu 14.11 (będzie m.in. o tym wtedy mowa, a następna pracownia, do której należy zrobić to zadanie, jest w odległym terminie 20.11), albo dla uproszczenia użyć zamiast tego tablicy zadeklarowanej globalnie.

Zadanie 3. Liczbę naturalną nazywamy k-pierwszą, jeżeli ma dokładnie k czynników w rozkładzie na czynniki pierwsze (licząc z powtórzeniami).

Na przykład:

k = 2 → 4 = 2^2 , 6 = 2×3 , 9 = 3^2 , 10 = 2×5 , ...

k = 3 → 8 = 2^3 , 12 = 2^2×3 , 18 = 2×3^2 , 20 = 2^2×5 , ...

k = 5 → 32 = 2^5 , 48 = 2^4×3 , 72 = 2^3×3^2 , 80 = 2^4×5 , ...

Liczby pierwsze są 1-pierwsze.

Napisz program, który:

wczytuje ze standardowego wejścia jedną dodatnią liczbę całkowitą n (n<40000) oblicza liczbę różnych rozwiązań równania: a + b + c = n

gdzie:
a jest liczbą 1-pierwszą (czyli liczbą pierwszą)
b jest liczbą 5-pierwszą (ma dokładnie 5 czynników pierwszych)
c jest liczbą 7-pierwszą (ma dokładnie 7 czynników pierwszych)

Wypisuje na standardowe wyjście znalezioną liczbę rozwiązań

Przykłady

A

Wejście:

```
376
```

Wyjście:

```
1
```

Jedyna znaleziona odpowidź to: 5 + 243 + 128

B

Wejście:

```
381
```

Wyjście:

```
11
```

C

Wejście:

```
222
```

Wyjście:

```
0
```