Sklep rybny (rozwiązanie)

Autor zadania: **Karol Pokorski**
Opracowanie: **Jakub Bartmiński, Karol Farbiś**
Opis rozwiązania: **Bartosz Kostka**

Olimpiada Informatyczna Juniorów

W zadaniu tym skupimy się na symulacji tego procesu. Dla każdej transakcji policzymy, ile maksymalnie kilogramów ryb możemy kupić, i wykonamy tę transakcję.

Jeżeli mamy N bajtalarów oraz M kilogramów ryb, a za 1 kilogram ryb musimy zapłacić jednego bajtalara i pół kilograma ryb, to maksymalna liczba kilogramów ryb, które możemy kupić (oznaczmy ją przez X), to minimum z liczby posiadanych bajtalarów oraz dwukrotności liczby kilogramów ryb, które posiadamy. Po transakcji liczba bajtalarów zmniejszy się o X, ilość ryb jednocześnie spadnie o $0.5 \cdot X$ (bo tyle musimy zapłacić), ale potem wzrośnie o X. Zatem ostateczny bilans dla liczby kilogramów ryb wyniesie $+0.5X$.

Zastanówmy się teraz, dlaczego symulowanie tych wszystkich operacji zmieści się w limicie czasu. Pokażemy, że faktycznych operacji nie będzie dużo. Zauważmy, że jeśli naszym ograniczeniem jest liczba posiadanych bajtalarów, a nie to, ile mamy kilogramów ryb, to natychmiast po wykonaniu jednej transakcji nie możemy kupić więcej ryb. Natomiast dopóki ogranicza nas liczba ryb, po każdej transakcji liczba kupionych ryb będzie rosnąć dwukrotnie. Dla przykładu, jeżeli początkowo mamy 100 bajtalarów, 2 kg ryb i nieograniczoną liczbę transakcji, transakcje będą wyglądały następująco:

- Kupujemy 4 kg ryb, płacąc 4 bajtalary i 2 kg ryb. Po tym mamy 96 bajtalarów i 4 kg ryb.
- Kupujemy 8 kg ryb, płacąc 8 bajtalarów i 4 kg ryb. Po tym mamy 88 bajtalarów i 8 kg ryb.
- Kupujemy 16 kg ryb, płacąc 16 bajtalarów i 8 kg ryb. Po tym mamy 72 bajtalary i 16 kg ryb.
- Kupujemy 32 kg ryb, płacąc 32 bajtalary i 16 kg ryb. Po tym mamy 40 bajtalarów i 32 kg ryb.
- Finalnie jesteśmy ograniczeni gotówką i kupujemy 40 kg ryb, płacąc 40 bajtalarów i 20 kg ryb. Finalnie mamy 0 bajtalarów i $32 - 20 + 40 = 52$ kg ryb.

To, jak szybko rośnie liczba kilogramów ryb, które możemy kupić, w informatyce nazywamy *wzrostem wykładniczym*. Możemy zauważyć, że jeżeli liczba bajtalarów spada za każdym razem o dwukrotnie większą wartość niż poprzednio, możemy oszacować, że liczba operacji nie przekroczy wykładnika, do jakiego należy podnieść liczbę 2, aby otrzymać liczbę bajtalarów posiadanych na początku. Jako że liczba bajtalarów nie przekroczy 10^{18}, a możemy sprawdzić, że $2^{60} > 10^{18}$, to liczba operacji nie przekroczy 60.

Pewną trudnością techniczną w tym zadaniu była reprezentacja liczb niecałkowitych. Niestety komputery bardzo często mają problemy z obliczeniami na takich liczbach. Dla przykładu prosimy sprawdzić następującą interakcję w interpreterze Pythona:

```
>>> X = 1000000000000000002
>>> print("%f" % (X/2))
500000000000000000.000000
>>> X / 2 == 500000000000000000
True
```

Widzimy tutaj, ku zaskoczeniu, że zwykłe podzielenie dużej liczby przez 2 może spowodować wypisanie niepoprawnego wyniku (i to nie o mało, aż o 1!).

Na szczęście w tym zadaniu możemy się tym zupełnie nie przejmować. Możemy pomnożyć wartości wszystkich ryb przez dwa. Możemy to interpretować w ten sposób, że wprowadzamy nową walutę *półkilorybie*, gdzie jedno półkilorybie jest równe pół kilograma ryb. Wtedy o wszystkich transakcjach możemy myśleć w następujący sposób: za dwa półkilorybia (jeden kg ryb) płacimy jednego bajtalara i jedno półkilorybie (0.5 kg faktycznej ryby). Wtedy wszystkie operacje możemy przeprowadzać na liczbach całkowitych. Jedynie ostateczną odpowiedź musimy ostrożnie podzielić przez dwa.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

ryb.cpp

```cpp
#include "bits/stdc++.h"

using namespace std;

int main() {
  long long N, M, K;
  cin >> N >> M >> K;

  // Mnożymy ilość ryb przez 2, aby wszystkie operacje
  // wykonywać na liczbach całkowitych.
  M *= 2;

  // Dopóki mamy zasoby (pieniądze, ryby)
  // oraz możemy wykonywać operacje (K)
  while (N > 0 && M > 0 && K > 0) {
    // Liczymy ile ryb możemy kupić.
    // Pamiętamy że ilość ryb jest podwojona.
    long long moge_kupic = min(N, M);

    // Aktualizujemy ile pieniędzy mamy.
    N -= moge_kupic;
    // Oraz ile ryb mamy po tej transakcji.
    // Pamiętamy że ilość ryb jest podwojona.
    M += moge_kupic;
    // Oraz zmniejszamy liczbę dostępnych operacji.
    K -= 1;
  }

  // Finalnie wypisujemy wynik w formacie z jedną
  // cyfrą po przecinku.
  cout << M / 2 << "." << (M % 2 == 0 ? "0" : "5") << "\n";

  return 0;
}
```

ryb.py

```python
def main():
  (N, M, K) = tuple(map(int, input().split()))

  # Mnożymy ilość ryb przez 2, aby wszystkie operacje
  # wykonywać na liczbach całkowitych.
  M *= 2

  # Dopóki mamy zasoby (pieniądze, ryby)
  # oraz możemy wykonywać operacje (K)
  while N > 0 and M > 0 and K > 0:
    # Liczymy ile ryb możemy kupić.
    # Pamiętamy że ilość ryb jest podwojona.
    moge_kupic = min(N, M)

    # Aktualizujemy ile pieniędzy mamy.
    N -= moge_kupic
    # Oraz ile ryb mamy po tej transakcji.
    # Pamiętamy że ilość ryb jest podwojona.
    M += moge_kupic
    # Oraz zmniejszamy liczbę dostępnych operacji.
    K -= 1
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".

```
22
23    # Finalnie wypisujemy wynik w formacie z jedną
24    # cyfrą po przecinku.
25    czesc_calkowita = str(M // 2)
26    czesc_ulamkowa = "0" if M % 2 == 0 else "5"
27    print(czesc_calkowita+"."+czesc_ulamkowa)
28
29  if __name__ == "__main__":
30      main()
```

Jako ciekawostkę, w języku Python istnieje możliwość, aby powiedzieć programowi, aby bardzo ostrożnie obchodził się z liczbami niecałkowitymi, używając biblioteki `decimal`. Przykładową implementację przedstawiamy poniżej. Należy być bardzo ostrożnym, używając tej biblioteki, ponieważ za dokładność oczywiście płacimy mocą obliczeniową, co w bardziej skomplikowanych obliczeniach może być powolne. W większości zadań nie polecamy używać tej biblioteki.

ryb_alt.py

```
1   from decimal import Decimal
2
3   def main():
4     # Uwaga implementacyjna:  aby utrzymac pewnosc
5     # ze liczby rzeczywiste beda poprawnie utrzymywane
6     # w pamieci, mozemy uzyc biblioteki decimal
7     # do otrzymywania zmiennych.
8     (N, M, K) = tuple(map(Decimal, input().split()))
9
10    # Dopóki mamy zasoby (pieniądze, ryby)
11    # oraz możemy wykonywać operacje (K)
12    while N > 0 and M > 0 and K > 0:
13       # Liczymy ile ryb możemy kupić.
14       moge_kupic = min(N, 2 * M)
15
16       # Aktualizujemy ile pieniędzy mamy.
17       N -= moge_kupic
18       # Oraz ile ryb mamy po tej transakcji.
19       M += moge_kupic / 2
20       # Oraz zmniejszamy liczbę dostępnych operacji.
21       K -= 1
22
23    # Finalnie wypisujemy wynik w formacie z jedną
24    # cyfrą po przecinku.
25    print("{:.1f}".format(M))
26
27  if __name__ == "__main__":
28      main()
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2022/2023, 2023/2024, 2024/2025".