mod 4

Напишете програма, която по зададени цели положителни числа x, y и n, изчислява по ефективен начин x^y mod n (операцията ^ означава повдигане на степен, а не изключващо или - XOR).

Жокер: Няма как да сметнем x^y в реално време при y = 2147483647.
Тази задача е от теория на делимост на числата. Тук важат следните правила:
(a+b) mod n = ((a mod n)+(b mod n)) mod n
(a*b) mod n = ((a mod n)*(b mod n)) mod n*
Също така ако имаме числата X, K и N, така че N не дели X ^ Q, и редицата E1(K) = (X^K) mod N то за дадено K = P <= N/НГОД(N,X) ще получим остатък, както за E1(1) = X mod N, а след това стойностите ще се повтарят ротационно. Това е логично, тъй като повторения остатък ще умножаваме по едно и също число X mod N (тъй като E1(P+1) = X^(P+1) mod N = (((X^P) mod N)(X mod N)) mod N = (((X^1) mod N)*(X mod N)) mod N = (E1(1)*(X mod N)) mod N), както при E1(1), за да получим следващия елемент от редицата.
Математически това изглежда така: E1(K) = E1(((K - 1) mod (P - 1)) + 1)

Имаме подобно правило (не касае тази задача) и при умножение на K и X: Ако имаме числата X, K и N и редицата E2(K) = (K.X) mod N за K = 1..(N/НГОД(N,X)) ще получим различни стойности без повторения за E2, а след това стойностите ще се повтарят ротационно.
Математически това изглежда така: E2(K) = E2(K mod (N/НГОД(N,X)))

Конкретната задача има три случая:

1) Ако n дели x, то резултатът е винаги 0.

2) Ако n дели x^q (пробваме за всяко q от 1 до n/НГОД(n,x) - може и до n ако имате достатъчно време за изпълнение на програмата и не ви се смята НГОД(n,x), като в тази задача), където q е естествено число: На колко ще е равно: x^y mod n при y < q (пресмятаме го докато проверяваме дали n дели x^q). А при y >= q (веднага можем да дадем математически отговор, тъй като в този случай x^y mod n = x^q.x^(y-q) mod n = ((x^q mod n) * (x^(y-q) mod n)) mod n = (??? * (x^(y-q) mod n)) mod n = ???

3) Ако n не дели x^q (всички останали различни от първия и втория случаи).
При кое число k ще е вярно : x^(1) mod n = x mod n = x^(1 + k) mod n
Това число го намираме програмно в стъпка 2 докато проверяваме дали n дели x ^ q.
На колко ще е равно: x^(1 + 2 * k) mod n
На колко ще е равно: x^(1 + 3 * k) mod n
...
На колко ще е равно: x^(1 + p * k) mod n
Където 1 + p * k е най-голямото цяло число по-малко или равно на y.
На колко ще е равно: x^y mod n
Знаейки колко е x^(1 + p * k) mod n

Има и още по-ефективен алгоритъм:
https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-

exponentiation
Прочетете го и се опитайте да го реализирате на C++ без да копи/паствате готов код от интернет.

Input Format

Първият ред на входа започва с числото C, следват C реда съдържащи числата x, y и n, разделени с интервал. Ред C + 1 съдържа числото 0.

Constraints

$0 < C < 40$, $1 < x, n < 2^{15} = 32768$, и $0 < y < 2^{31} = 2147483648$.

Output Format

Изходът трябва да съдържа C реда, като i-тия ред се намира положителното число z, такова че $z = x^y \bmod n$ за зададените в i-тия тест x , y и n.

Sample Input 0

```
2
2 3 5
2 2147483647 13
0
```

Sample Output 0

```
3
11
```