A processzorok bitműveletek segítségével sokszor gazdaságosabban és gyorsabban végzik el két egész szám szorzását, mint más módon. Tízes számrendszerben egy szám végére 0-t írva annak 10-szeresét kapjuk, míg kettes számrendszerben az eredeti szám 2-szeresét. Ha tehát egy bináris számot eltolunk 3-mal a nagyobb helyiértékek felé, miközben a szám végére három 0-át írunk, akkor egy 8-cal történő szorzást végzünk. Ha a kapott értékhez még hozzáadjuk az eredeti számot, akkor valójában 9-cel szorzunk.

Ezek alapján minden egész számmal történő szorzás megvalósítható bizonyos számú eltolás, a közben kapott értékek tárolása, és valahány összeadás segítségével. Például az $x \cdot 29$ fölírható

$$x \cdot (28+1) = x \cdot (2 \cdot 14 + 1) = x \cdot (2 \cdot 2 \cdot 7 + 1) = x \cdot \big(2 \cdot 2 \cdot (6+1) + 1\big) =$$
$$= x \cdot \big(2 \cdot 2 \cdot (2 \cdot 3 + 1) + 1\big) = x \cdot \big(2 \cdot 2 \cdot \big(2 \cdot (2+1) + 1\big) + 1\big)$$

alakban. Ha ezt fölbontjuk, akkor az $x \cdot 2 \cdot 2 \cdot 2 \cdot 2 + x \cdot 2 \cdot 2 \cdot 2 + x \cdot 2 \cdot 2 + x$ kifejezést kapjuk, ahol csak 2-vel való szorzás (tehát eltolás), illetve összeadás szerepel.

Tegyük föl, hogy a részeredmények tárolásához elegendő hely áll rendelkezésre. Adjuk meg, hogy ezzel a módszerrel végezve hány eltolás és hány összeadás szükséges egy adott számmal történő szorzás elvégzéséhez. A 29-cel való szorzáshoz például ki kell számítanunk az $x \cdot 2$, $x \cdot 2 \cdot 2$, $x \cdot 2 \cdot 2 \cdot 2$, $x \cdot 2 \cdot 2 \cdot 2 \cdot 2$ értékét, amelyekhez összesen 4 eltolás szükséges, és ezután kell még három összeadás.

A program a standard bemenet első sorából olvassa be a szorzót (pozitív egész), majd írja ki a standard kimenet egyetlen sorába elsőként az eltolások számát, majd szóközzel elválasztva az összeadások számát.

Beküldendő egy `i496.zip` tömörített állományban a program forráskódja és egy rövid leírás, ami megadja, hogy a forrásállomány melyik fejlesztői környezetben fordítható.