Ha $N \leq 0$, a rutin A-ba B értékétől függetlenül 1-et ír. Különben N-en addig végez 2-vel való egész osztást, amíg az 0-ra csökken. Közben minden lépésben B aktuális tartalmát négyzetre emeli. Ha tehát $2^k \leq N < 2^{k+1}$, akkor a program $k+1$ lépést végez, és végül is B-be az eredeti érték 2^{k+1}-edik hatványa kerül. Vizsgáljuk meg, mi történik közben A-val. Valahányszor N aktuális értéke páratlan, megszorzódik B aktuális értékével. Az eljárás tehát az 1734. gyakorlat szorzására emlékeztet, csak most összeadás helyett szorzás az elemi lépés. Ennek megfelelően az eredmény B eredeti értékének N-edik hatványa (N-ben is az eredeti értéket véve). A bizonyítás lényegében ugyanaz, mint a gyakorlat esetében, így nem részletezzük. Azt, hogy a program azt számolja-e ki, amit kell, nyilván csak a készítője mondhatja meg. Nem valószínű azonban, hogy szükség van B és N értékének megváltoztatására. Szerencsésebb megoldásnak tűnik, ha az első sort a következő háromra cseréljük:

```
                                                ccc
SUBROUTINE SS(A,BB,NN)  |
B=BB                    |
N=NN.                   |
```

Sali Attila (Budapest, Fazekas M. Gyak. Gimn., IV. o. t.)

Megjegyzés. A program aránylag kevés művelettel állítja elő A N-edik hatványát. Mint láttuk, ehhez N szorzás helyett legfeljebb $5(\log_2 N + 1)$ műveletre van szüksége. Hasonló feladat az $(A_I, B_I, I = 1, 2, \ldots, N)$ vektorokból a

$$C_K = \sum_{I=1}^{K} A_I B_{K-I+1}, \quad K = 1,\ 2,\ \ldots,\ N$$

vektor előállítása (ez lényegében hosszú számok szorzásának felel meg). Itt is van a triviális N^2 lépésnél lényegesen gazdaságosabb megoldás: nagyságrendileg $N \log N$ lépés is elegendő.

Nem tudjuk azonban, hogy ennél lényegesen kevesebb lépés nem elegendő-e a művelet végrehajtásához.