Makra používaná v programech:

```
# skočí na návěstí N
MAKRO skoč N
   přenes [nová_prázdná_lokalita] J K N
KONEC
```

a Nejprve do proměnné jáma přidáme z kamenolomu to samé, co v ní již je. Tím hodnotu zdvojnásobíme. Poté tento dvojnásobek opět nabereme v kamenolomu a vyložíme v jámě. Získáme tím dvojnásobek dvojnásobku původní hodnoty, tedy celkově čtyřnásobek původní hodnoty

```
přenes K jáma jáma -
přenes K jáma jáma -
```

b Nejprve přeneseme do C tolik kamenů, kolik je v A. Poté se pokusíme odečíst od C tolik, kolik je v B. Jestliže se to nepovede, v B je více a končíme program. Jestliže se to povede, může být v obou stanovištích stejně kamenů, nebo v A více. To zjistíme odečtením 1 od C. Jestliže to nejde, jsou stejné, uložíme tam 1 kámen a končíme. Jestliže to jde, je v A více, nulujeme a končíme.

```
          přenes K A C -
          přenes C B K nulování # → B > A
          přenes C J K ulož1 # → B = A
          skoč nulování
          skoč konec
nulování: přenes C C K -
          skoč konec
ulož1:    přenes K J C -
          skoč konec
```

c Jelikož nám vůbec nezáleží na časové složitosti, navrhneme nejjednodušší řešení, při kterém je výsledek dělení počet možných odečtení dělitele od dělence. Jako zbytek poté považujeme číslo, které zbylo z dělence po posledním odečtení. V normálním programovacím jazyce bychom to napsali takto:

```
void podilSeZbytkem(int a, int b, int c, int d){
   int d = a;
   while(d - b > 0){
      d -= b
      c++;
   }
   #c = výsledek podílu, d = zbytek
}
```

Po transformaci dostaneme:

```
MAKRO deleniSeZbytkem A B C D
          přenes K A D -
   cyklus: přenes D B K konec
          přenes K J C -
          skoč cyklus
KONEC
```

d Pro tuto úlohu využijeme Euklidův algoritmus, který můžeme napsat například takto:

```
int nsd(int a, int b){
   int c;
   while(b != 0){
      c = a % b;
      a = b;
      b = c;
   }
   #Výsledek je v proměnných b, c
}
Využijeme předchozí úlohy a napíšeme si nejprve makro pro pouhé modulování

\begin{verbatim}
MAKRO modulo A B C
   přenes C C K - #nulování, C nemusí být na začátku nula
   přenes K A C -
   cyklus: přenes C B K konec
           skoč cyklus
KONEC
```

Po transformaci získáme:

```
MARKO nsd A B C
   cyklus: modulo A B C
           přenes A A K - #nulování
           přenes B B A -
           přenes B B K - #nulování
           přenes C C B -
           přenes B J K konec # b=0
           přenes K J B -
           skoč cyklus
KONEC
```