Az internetes csomagkapcsolt kommunikáció üzenetküldő eljárását az alábbi egyszerű modellel vizsgáljuk.

A kommunikációt létrehozó szoftverek a kommunikálók egymásnak szánt üzeneteit szabvány méretű darabokra tördelik, azokat a továbbításhoz szükséges adatokkal egészítik ki. Ezeket, az úgynevezett csomagokat küldik el. A csomagok nem feltétlenül a feladás sorrendjében érkeznek meg a címzetthez.

Az egyes üzenetdarabokat tartalmazó csomag szerkezete a következő:

feladó	a feladó azonosítója, pozitív egész, értéke legfeljebb 100;
címzett	a címzett azonosítója, pozitív egész, értéke legfeljebb 100;
üzenetazonosító	egyedi, értéke legfeljebb 1 000 000;
részletsorszám	megadja, hogy az adott üzenet melyik darabjáról van szó, a részletek sorszámozása folyamatos, az első értéke 1;
üzenetrészlet	pontosan 10 karakter. Az üzenet utolsó értékes karakterét legalább egy, de mindig pontosan a 10 karakter kitöltéséhez szükséges számú szűrő karakter (|) követi. (Az üzenet csak az angol ábécé kis és nagybetűiből, számjegyekből, írásjelekből, kötőjelből és szóköz karakterből áll.) A szóköz karakter helyett aláhúzás karaktert használunk a könnyebb kezelhetőség érdekében;
ellenőrző összeg	képzési módja a következő: az első négy számadat és az üzenetdarab karaktereinek ASCII kódjából képzett összeg 100-as osztási maradéka.

A hálózati modell működését egy rövid teszttel vizsgáljuk, melynek részeként az összes beérkező üzenetdarabot a `naplo.txt` fájlba rögzítjük.

Például:

`naplo.txt`

```
11 3 334 2 verseny||| 2

3 2 417 1 Hatarido:\_88

11 3 334 1 KoMaL\_pont 29

...
```

Az 1. sor megmutatja, hogy a 11. állomás küldött üzenetet a 3. állomásnak. Az üzenetet a 334-es azonosítóval látta el, amelynek második részét tartalmazza ez a csomag. Mivel a szűrő karakter (|) szerepel, ezért ez az üzenet utolsó darabja.

Készítsünk programot `halo` néven, amely az alábbi kérdésekre válaszol. Ügyeljünk arra, hogy a program minden helyes tartalmú bemeneti állomány esetén működjön.

Minden részfeladat megoldása előtt írjuk a képernyőre a feladat sorszámát. Ha a felhasználótól kérünk be adatot, jelenítsük meg a képernyőn, hogy milyen értéket várunk (például az 5. feladat esetén: „5. feladat: Adja meg az üzenet szövegét!”). A képernyőn megjelenített üzenetek esetén az ékezetmentes kiírás is elfogadott.

1. Olvassuk be a `naplo.txt` állományban talált adatokat (honlapunkról egy mintaállomány letölthető), s azok felhasználásával oldjuk meg a következő feladatokat. Ha az állományt nem tudja a programmal beolvasni, az első 10 csomaghoz tartozó adatokat jegyezzük be a programba és dolgozzunk azzal.

2. Jelenítsük meg a képernyőn, hány csomagot küldött az 4-es állomás az 5-ösnek.

3. Írjuk a képernyőre a hibás ellenőrző összeget tartalmazó sorok sorszámát. Az egyes értékeket szóközzel határoljuk. Ha nem volt ilyen, akkor a „Nem volt hibás ellenőrző összeg.” szöveget jelenítsük meg.

4. A `naplo.txt` állomány feldolgozásával állítsuk elő az eredeti üzeneteket. Az `eredeti.txt` állományba soronként egy-egy üzenetet jegyezzünk be. A fájl tartalmát az alábbi mintának megfelelően alakítsuk ki. (A minta illeszkedik a feladat elején megadott bemenethez.) Az üzeneteket tetszőleges sorrendben megadhatjuk.

```
eredeti.txt
        11 3 334 KoMaL\_pontverseny
        ...
```

5. Bontsunk csomagokra egy üzenetet. A felhasználótól kérjük be az üzenet szövegét, valamint azt, hogy melyik állomástól melyikhez kell eljuttatni. Az elkészített csomagokat soronként írjuk a képernyőre. Biztosítsuk, hogy az üzenet azonosítója ne egyezzen meg a korábbi üzenetek egyikével sem, de egyébként tetszőleges érték lehet.

Beküldendő egy tömörített (`i232.zip`) állományban a program forráskódja (`i232.pas`, `i232.cpp`, ...), valamint a program rövid dokumentációja (`i232.txt`, `i232.pdf`, ...), amely tartalmazza a megoldás rövid leírását, és megadja, hogy a forrásállomány melyik fejlesztő környezetben fordítható.