Adott a következő struktúra:

```
struct teglalap {
  int szel;
  int mag;
};
```

ami egy téglalap szelességét (szel) és magasságát (mag) határozza meg.

Definiálja azt a void nyomtat(const teglalap& t) függvényt, ami kirajzolja a karakteres kimenetre a téglalapot úgy, hogy annak négy sarkát + jellel jelöli, a vízszintes oldalait - jelekkel, a függőleges oldalait pedig | jelekkel rajzolja meg!

For example:

Test Result

```
teglalap t1 = { 5, 3 };
nyomtat(t1);
```

```
+---+
|   |
+---+
```

Az alsó háromszögmátrix (vagy triangulummátrix) olyan négyzetes mátrix, melynek a főátlója feletti összes elem zéró. Formálisan: ui,j=0, ha i<j.

Fejezze be az alábbi l függvényt, melynek paramétere egy matrix típusú struktúrát címző mutató. A struktúra szerkezete a következő:

```
struct matrix {
  int sorok; // A matrix sorainak szama
  int oszlopok; // A matrix oszlopainak szama
  double** m; // Matrix mutatotombos alakban megadva; 'sorok' darab 'double*' mutato, ami 'oszlopok' elemu 'double' tombot cimez.
};
```

A függvény visszatérési értéke legyen true, ha a paraméter alsó háromszögmátrix, különben false.

Szintaktikai hiba esetén a fordító által jelzett sor számából vonjon ki 20-at, hogy megkapja a szerkesztőben olvasható sorszámot!

Készítse el az alábbi cink függvényt, mely egy csalásra előkészített 6 oldalú dobókockával történő dobást szimulál, azaz visszatérési értéke egy véletlenszerűen választott egész az [1, 6] intervallumból, de a 6-os dobás valószínűsége 13, míg az összes többi számé egyformán csak 215.

Feltételezheti, hogy a véletlenszám-generátort előzetesen már inicializálták.

For example:

Test	Result
srand(time(NULL)); cout << (ellenoriz()?"OK":"Hibas megvalositas") << endl;	OK

Az ISBN egy azonosítószám, a könyvek és egyéb monografikus jellegű művek nyilvántartására szolgáló nemzetközi szabványos számrendszerhez tartozó kód. Készítse el az isbn függvényt, ami igaz értéket ad vissza, ha a string típusú paraméterként átadott adat egy formailag helyes ISBN, egyébként hamisat!

Egy kód akkor helyes, ha csupa számjegy karakterből áll, hossza 10 vagy 13 karakter, és az utolsó helyiértéken szereplő ellenőrző összeg is megfelelő. Az ellenőrző összeg számítása az eltérő hosszú kódok esetén eltér.

A 10 számjegyű ISBN esetén először azt a szorzatösszeget kell kiszámolni, amit az első 9 számjegy 10-től 2-ig csökkenő súlyokkal történő beszorzásával kapunk. Az ellenőrző összeg értéke az a szám, mely az iménti összeg 11-gyel végzett osztásának maradékát 11-re egészíti ki.

Pl., ha az ISBN első 9 karaktere 030640615, akkor

10×0 + 9×3 + 8×0 + 7×6 + 6×4 + 5×0 + 4×6 + 3×1 + 2×5

= 0 + 27 + 0 + 42 + 24 + 0 + 24 + 3 + 10

= 130

11 legközelebbi többszöröse: 12×11 = 132

132 - 130 = 2, ez tehát az ellenőrző összeg, aminek a 10. helyiértéken szerepelni kell.

A 13 számjegyű ISBN esetén az ellenőrző összeg számítása úgy történik, hogy először az első 12 számjegyből számolnak szorzatösszeget, de most a súlyok váltakozva mindig csak 1 és 3. Ezt az összeget osztják maradékosan 10-zel, és az eredményt 10-re kiegészítendő szám lesz az ellenőrző összeg. Ez adódhat 10-nek is, amit 0-val helyettesítenek, mivel erre a célra csak 1 számjegy áll rendelkezésre a kódban.

Pl. ha az első 12 számjegy 978030640615, akkor

9×1 + 7×3 + 8×1 + 0×3 + 3×1 + 0×3 + 6×1 + 4×3 + 0×1 + 6×3 + 1×1 + 5×3

= 9 + 21 + 8 + 0 + 3 + 0 + 6 + 12 + 0 + 18 + 1 + 15

= 93

93 / 10 = 9 maradék 3

10 – 3 = 7 az ellenőrző összeg.

Adott a következő struktúra:

```
struct elem {
  string szoveg;
  elem* alpontok;
  elem* kov;
};
```

Ezzel egy többszintű lista (nem számozott felsorolás) elemeit lehet tárolni. A szoveg adja meg az aktuális felsorolt elem szövegét, kov pedig a következő felsorolt elem helyét a tárban (ami NULL értékű, ha nincs több elem). A felsorolás elemei egy irányban láncolt listát alkotnak. Az alpontok az aktuális elem alatti szinten lévő első felsorolt elemet jelöli ki, azaz egy beágyazott láncolt lista elejét.

Definiálja azt a void nyomtat(const elem* horgony, int szint = 0) függvényt, ami kinyomtatja a teljes felsorolást, ha a legkülső lista horgonypontjával hívják meg. Rekurzívan hívnia kell önmagát, amennyiben az aktuális elemnek alpontjai vannak, melyek szint-je mindig eggyel nagyobb az aktuálishoz képest. A felsorolt elemek elé nyomtasson ki annyi szóközt, mint a szint értékének kétszerese (ez igazítja az elemeket), majd egy kötőjelet és egy szóközt is az elem szövege elé!

For example:

Test Result

```
elem* horgony = NULL;
beszur(horgony, "Harom");
elem* ketto = beszur(horgony, "Ketto");
beszur(ketto->alpontok, "Ketto.Harom");
beszur(ketto->alpontok, "Ketto.Ketto");
beszur(ketto->alpontok, "Ketto.Egy");
beszur(horgony, "Egy");
nyomtat(horgony);
felszabadit(horgony);
```

```
- Egy
- Ketto
  - Ketto.Egy
  - Ketto.Ketto
```

- Ketto.Harom

- Harom