1

Írjon egy programot, amely 2 db felhasználói függvényt használ.
Az egyik az **unsigned int beker(int be_tb[])**, amely 10 db előjeles egész számot kér be úgy,
hogy azokat egy tömbbe tárolja le (1. paraméter). A tömb hosszát szimbolikus állandóval állítsa be.
A tárolás csak akkor jöjjön létre, ha az első 5 elem pozitív, amíg a második 5 elem negatív.
Ha az adott érték nem teljesíti a rá vonatkozó feltételt, akkor új számot kell bekérni.
A bekérési folyamatnak addig kell folytatódnia, amíg a tömb teljes feltöltése meg nem történik.
Bekérési információk és hibaüzenetek kiírása nem kell!
A feltételek teljesülése esetén a **beker** függvény határozza meg külön-külön a pozitív, illetve a negatív számok szorzatát.
Ezek után az abszolút értékben kisebb szorzattal térjen vissza (visszatérési érték).
A másik függvény a **void kiir(unsigned int sr)**, pedig írja ki a **beker** függvény visszatérési értékét
(1. paraméter) a példában látott szövegezéssel.

For example:

Input	Result
`1 2 3 4 5 -6 -7 -8 -9 -10`	`A kisebb szorzat: 120`
`6 7 8 9 10 -1 -2 -3 -4 -5`	`A kisebb szorzat: 120`

```
#include<iostream>

unsigned int beker(int be_tb[])
{
 . . .
}

void kiir(unsigned int sr)
{
 . . .
}

int main()
{
  . . .
return 0;
 }
```

2

Írjon egy programot, amely 2 db felhasználói függvényt használ.
Az egyik az int feltolt(char s_tb[]), amely feltölt egy karakter tömböt (1. paraméter) ékezet nélküli szöveggel az ENTER billentyű leütéséig, vagy maximum 50 db karakterig.
Bekérési információk és hibaüzenetek kiírása nem kell! A tömb hosszát szimbolikus állandóval állítsa be.
A függvény visszatérési értéke a feltöltött tömb hossza legyen.
A másik függvény a void cserel(char s_tb[], int h), pedig a feltöltött tömb (1. paraméter),
elemeit cserélje fel páronként. Azaz:
- az 1. elemet a 2.-kal,
- a 3. elemet a 4.-kel,
- az 5. elemet a 6.-kal és így tovább.

Ha páratlan számú a karaktersorozat, akkor az utolsó karakter maradjon a helyén, hiszen nincsen párja! (Ilyen a második és a harmadik példa is.)
Ezek után a fentiek szerint *módosított* tömb tartalmát írja is ki.
A cserel függvény 2. paramétere a feltolt függvény visszatérési értéke.

For example:

Input	Result
`alma korte szilvas`	`laamk roets izvlsa`
`alma korte szilva`	`laamk roets izvla`
`1a2b3c4d5e6f7g8h9`	`a1b2c3d4e5f6g7h89`

```
#include<iostream>
#include <cstring>

int feltolt(char s_tb[])
{
  ...
}

void cserel(char s_tb[], int h)
{
 ...
}

int main()
{
 ...
```

```
 return 0;
}
```

3. Írjon egy string run_result(string fbe) függvényt, amelynek a visszatérési értéke egy 3 körös futóverseny legjobb köridejét teljesítő versenyző azonosítója (licence) legyen.
A versenyző bármelyik körben elérheti ezt az időt.
(A feladat könnyítése érdekében tételezzük fel, hogy csak 1 db *legjobb* köridő van!)
A célba érkezett versenyzők számát és a köridőket egy adat-file tartalmazza, amelynek a létezését ellenőrizni kell! A hibaüzenet formátumát a példa is mutatja!
Ennek az állománynak az azonosítója lesz a run_result függvény paramétere.
Az adat-file első sora egy pozitív egész szám, amely a célba érkezettek száma.
A további mindenegyes sora egy-egy sportolót azonosít és tartalmazza a köridejeit másodpercben, a következők szerint:

licence 1._köridő 2._köridő 3._köridő

Az adatokat szóközök választják el egymástól! Pl.:

df-572ki 1100 1101 1102
...
do-565yy 1100 1001 1002

Az adat-file elemeit tárolja el egy struktúra-tömbbe, amihez használja a megadott struct runner típust! A struktúra-tömböt a dinamikus memóriába hozza létre!
A versenyzők legjobb köridejeit az int best tagváltozókba mentsék el.
Ezen adatokból kell megállapítani, hogy ki futotta meg a verseny leggyorsabb körét.
Ha az adat-file nem létezik, akkor a visszatérési érték a "Sikertelen file-nyitás!" karaktersorozat legyen!
(A cerr << utasítást NE használják! moodle...)

For example:

Test	Result
`string best=run_result("Chip.txt");` `std::cout << best;`	`kt-942uu`
`string best=run_result("NoChip.txt");` `std::cout << best;`	`Sikertelen file-nyitás!`

```
#include<iostream>
#include<fstream>

struct runner {
  string lic;
  int run_1;
  int run_2;
  int run_3;
  int best;
};

string run_result(string fbe)
```

```
{
  ...
}
```

4. Írjon egy double** befogok(string fbe) függvényt, amelynek a visszatérési értéke egy
10 x 2-es dinamikus mátrix, amely 10 db derékszögű háromszög befogóit tartalmazza,
a lenti meghatározások alapján.
A befogók kiszámításához szükséges adatokat egy adat-file tartalmazza, amelynek a létezését
ellenőrizni kell! A hibaüzenet formátumát a példa mutatja!
Ennek az állománynak az azonosítója lesz, a befogok függvény paramétere.
Az adat-file egy-egy sora egy derékszögű háromszög kettő adatát tartalmazza, ezek az:

átfogó és_az_egyik_hegyesszög

Az adatokat szóköz választja el egymástól! Az átfogó mértékegysége méter, a szögé fok. Pl.:

140 40
...
120 15

Az adat-file soronkénti elemeit szintén egy 10 x 2-es dinamikus mátrixba tárolja el és ezen adatokból számítsa ki a befogókat, a következő képletek segítségével:
- a hegyesszöggel szembeni befogó = átfogó * sin(hegyesszög),
- a hegyesszög melletti befogó = átfogó * cos(hegyesszög).

A befogókat ebben a sorrendben kell tárolni a visszatérési értékként kezelt double mátrixban!
A dinamikus tömbök méretét szimbolikus állandókkal állítsa be.
A befogók tizedespontosságának a meghatározását bízzák a fordítóra.
(A cerr << utasítást NE használják! moodle...)

For example:

Test	Result

`double** bf=befogok("Tri.txt");`
`if(bf) {`
`for(int i=0; i<10; i++) {`
`cout << bf[i][0] << " " << bf[i][1] << endl; }`
`for(int i=0; i<10; i++) { delete [] bf[i]; bf[i]=0; }`
`delete [] bf; bf=0; }`
	`89.9903 107.246`
	`70 121.244`
	`32.1394 38.3022`
	`25 43.3013`
	`57.4533 48.2091`
	`64.9519 37.5`
	`55 95.2628`
	`70.7066 84.2649`
	`84.8528 84.8528`
	`31.0583 115.911`

`double** bf=befogok("NoTri.txt");`
`if(bf) {`
`for(int i=0; i<10; i++) {`
`cout << bf[i][0] << " " << bf[i][1] << endl; }`
`for(int i=0; i<10; i++) { delete [] bf[i]; bf[i]=0; }`
`delete [] bf; bf=0; }`
	`Sikertelen file-nyitás!`

```
#include<iostream>
#include<fstream>
#include<cmath>

double** befogok(string fbe)
{
 ...
}
```

5.Írjon egy string** first_last(string fbe, string fki) függvényt, amelynek a visszatérési értéke egy dinamikus mutató tömb, amely egy olyan, szintén dinamikus tömbre mutat,
amelyben az alábbi átalakításokon átesett szerzők nevei találhatóak.
Sőt, az átalakított szerző neveket ki kell menteni egy kimeneti file-ba (2. paraméter) is.
A kimeneti file létrejöttét is ellenőrizni kell! Az átalakítás pontos leírását alább olvashatja.
Az átalakítandó szerzők keresztneveit (több is lehet) és vezetéknevét egy adat-file tartalmazza,
amelynek a létezését ellenőrizni kell! A hibaüzenet formátumát a példa mutatja!
Ennek az állománynak az azonosítója lesz az first_last függvény 1. paramétere.
DE! Az adat-file első sorában csak egy pozitív egész szám található, amely az írók számát
adja meg, amíg a további sorok egy-egy szerzőt azonosítanak a következőképpen:

1._keresztnév 2._keresztnév ... n._keresztnév vezetéknév

A neveket szóközök választják el egymástól! Pl.:

Douglas Noel Adams
...
Timothy Zahn

Az adat-file szerző-sorait tárolja el egy dinamikus 2 dimenziós tömbbe.
Ezek után dolgozza fel ezen tömböt úgy, hogy a szerzők nevei a következő mintát kövessék, mind a visszatérési érték által mutatott tömbben, mind a kimeneti file-ban:

vezetéknév, 1._keresztnév 2._keresztnév ... n._keresztnév

Azaz a vezetéknév kerüljön előre, majd a vessző utáni szóköz után jöjjenek a keresztnevek szóközökkel elválasztva.

For example:

Test	Result

`string** vkn=first_last("Writers.txt","Sretirw.txt");`
`if(vkn) {`
`  int rs=sizeof(vkn);`
`  cout << rs << endl;`
`  for(int j=0; j<14; j++) { cout << vkn[rs-1][j]; }`
`  for(int i=0; i<rs; i++) { delete [] vkn[i]; vkn[i]=0;`
`}`
`  delete [] vkn; vkn=0;`
`  string ide;`
`  ifstream fg("Sretirw.txt");`
`  if(fg.is_open()) {`
`   getline(fg,ide);`
`   cout << ide << endl; fg.close(); }`
`  else cout << "Nem hozott létre kimeneti file-t!"; }`
	`8`
	`Zahn, Timothy`
	`Adams, Douglas Noel`

Test	Result

`string** vkn=first_last("NoWriters.txt","Sretirw.txt");`
`if(vkn) {`
`  int rs=sizeof(vkn);`
`  cout << rs << endl;`
`  for(int j=0; j<14; j++) { cout << vkn[rs-1][j]; }`
`  for(int i=0; i<rs; i++) { delete [] vkn[i]; vkn[i]=0;`
`}`
`  delete [] vkn; vkn=0;`
`  string ide;`
`  ifstream fg("Sretirw.txt");`
`  if(fg.is_open()) {`
`   getline(fg,ide);`
`   cout << ide << endl; fg.close(); }`
`  else cout << "Nem hozott létre kimeneti file-t!"; }`
	`Sikertelen file-nyitás!`

```
#include<iostream>
#include<fstream>
#include<cstring>

string** first_last(string fbe, string fki)
{
 ...
}
```