(Akinek nem sikerült múltkor végigérnie a feladatsoron az kezdheti az 5. gyakorlat utolsó (stringes) feladatával, onnan legalább az els? pöttyöt mindenkinek meg kell tudnia oldani, és a ZH-hoz nem fog ártani a stringek kezelésének gyakorlása.)

String-összehasonlítás

Írj függvényt ami megkap két stringet (mutatókkal), és 1-et ad vissza ha az els? a nagyobb (lexikografikus rendezés, vagyis ABC-rend szerint), -1-et ad ha a második a nagyobb, és 0-t ad ha egyforma a két string. A kapott stringeket még véletlenül se változtassa meg a függvény, ezért legyenek konstansok a paraméterek:

```
int nagyobb_str(const char *s1, const char *s2) {
...
}
```

A függvény megírása után teszteld is néhányféleképpen a main() függvényedb?l meghívva az összehasonlítót. Pl:

```
char egyik[20] = "abrakadabra";
char masik[10] = "sotetkek";
int n = nagyobb_str(egyik, masik)
```

Ezután *n*-nek -1 érték?nek kell lennie mert a másik string a nagyobb.

Kétirányba láncolt lista

Írj programot, ami megvalósít egy kétirányba láncolt listát! Kiindulhatsz az el?adáson vett kódból (ott egy irányban volt csak megoldva a láncolás). Amiken módosítani kell:

- A listaelemekben legyen az azonosítón kívül egy max. 40 karakteres "http://wiki.math.bme.hunev"http://wiki.math.bme.hu is eltárolva
- Kétirányúsítás: a listaelemekben legyen még egy pointer ami az el?z? elemre mutat a listában (a lista el? eleménél ez NULL)
- Legyen egy függvény ami a lista utolsó elemét adja vissza (A lista struktúrában legyen mutató az utolsó elemre is)
- Írj függvényt ami kap egy listát és egy azonosítót, és visszaadja az azonosítóhoz tartozó nevet ha talált ilyen azonosítót a listában (elölr?l keressen, az els? találtaot adja vissza, ha nem talált semmit akkor üres stringet)
- Írj egy "http://wiki.math.bme.huuj_elem()"http://wiki.math.bme.hu függvényt ami megkap egy long azonosítót és egy (max 40 karakteres) char* nevet, *malloc()*-cal dinamikusan helyet foglal egy új elemnek, a pointereit NULL-ra állítja és visszaadja az új elemre mutató pointert.
- Írj egy beszúró függvényt ami a kapott listába egy elemet (amit egy pointerrel kap meg), beszúr arra a helyre ahová név szerint rendezve be kell tenni. Haszáld az el?z? feladatben megírt függvényt (vagy a string.h-ból az "http://wiki.math.bme.hustrcmp()"http://wiki.math.bme.hu-t). Ha üres volt eddig a lista (a kezd?elem és az utolsó elem pointer is NULL) akkor persze csak vedd fel ezt az elemet els?ként és utolsóként is. Ha nem volt üres a lista, akkor figyelj arra hogy minden pointert jól átállíts! A beszúrt elem el?tti elem "http://wiki.math.bme.hukovetkezo"http://wiki.math.bme.hu mutatója az új elemre mutasson, és a beszúrt elem utáni elem "http://wiki.math.bme.huelozo"http://wiki.math.bme.hu pointere is az új lemere mutasson (eddig ezek egymásra mutattak). És persze az újonnan beszúrt elem mutatóit is be kell állítani.
- A main() függvényben egy for ciklusban kérj be a felhasználótól (egyszavas) neveket és long típusú azonosítókat (4 darabot), és a rendezve beszúró függvénnyel szúrd be ?ket a listába. Végül írasd ki a lista elemeit a láncolás szerinti sorrendben.