Az eheti házi az 5. gyakorlat utolsó 2. feladatára, a betûraktárra épül, ennek a megoldását letölthetitek innen. Négy függvénnyel kell kiegészítenetek a programot, a függvények a megadott deklarációkkal (ezek legyenek a függvényeitek feje):

- *char* keres_torol(char s)*: az els? polctól és els? doboztól kezdve végigkeresi a raktárat és törli ('\0'-ra állítja) az els? olyan karaktert aminek az értéke megegyezik a kapott *s* karakterrel, majd adja vissza a pointerét. Ha nem találja a keresett elemet a raktárban, akkor NULL-t adjon vissza. Piros pontért (és rendesen le is rövidíti a kódot), használjátok a *holvan* függvényt a megvalósításhoz.
- *int azonosak(char s)*: visszaadja, hogy hány darab olyan értékû karakter van a raktárban mint a kapott *s*.
- *void tomb_betesz(char t[], int n, int polc)*: a függvénynek az a célja, hogy az adott *polc*-ra, bepakolja a *t* tömbben tárolt *n* darab karaktert üres helyekre. A függvény vizsgálja, hogy az adott *polc* létezik-e, ha nem akkor kiír egy hibaüzenetet. Továbbá, a függvény vigyáz arra is, hogy ha nincs elég hely az adott polcon, akkor nem is kezdi el betenni a karaktereket (használjátok a *polcon_darab* függvényt) és ebben az esetben is írjon hibaüzenetet. A függvény megírásához hasznos lehet az *ures_helyre_pakol* függvény.
- *int keres_mind_torol(char s)*: végigkeresi a raktárat és törli ('\0'-ra állítja) az összes olyan karaktert aminek az értéke megegyezik a kapott *s* karakterrel, majd visszaadja, hogy mennyit törölt.
  SPOILER: használjátok az *azonosak* függvényt, hogy megtudjátok hány darab ilyen karakter van a raktárban, majd hívjátok meg ennyiszer a *keres_torol*-t.

A teszteléshez használhatjátok ezt a main függvényt:

```
int main(void){
    int i;
    char t[] = {'e', 'e', 't', 'e', 'e'};
    raktar  = (char**)malloc(polcok * sizeof(char*));
    for(i = 0; i < polcok; i++){
        raktar[i] = (char*)malloc(dobozok * sizeof(char));
    }
    urit();
    betesz('g', 2, 5);
    betesz('e', 6, 6);
    betesz('v', 4, 2);
    betesz('b', 2, 3);
    betesz('h', 5, 7);
    betesz('j', 7, 6);
    betesz('r', 4, 2);
    betesz('e', 2, 8);
    betesz('j', 2, 7);
    betesz('t', 2, 4);
    betesz('w', 1, 7);
    betesz('y', 4, 5);
    betesz('e', 6, 3);
    betesz('g', 1, 2);
    betesz('z', 2, 1);
    printf("azonosak test: %d\n", azonosak('e')); // 3
    keres_torol('e');
    printf("keres_torol test: %d\n", azonosak('e')); // 2
    keres_mind_torol('e');
    printf("keres_mind_torol test: %d\n", azonosak('e')); // 0
    tomb_betesz(t, 5, 12);
    tomb_betesz(t, 5, 2);
    tomb_betesz(t, 5, 3);
    printf("tomb_betesz test: %d\n", azonosak('e')); // 4
    return 0;
}
```