1. Adott a következő típusdefiníció:

```
struct tag{int adat; struct tag * bal, * jobb;}
```

   Írjon függvényt, mely visszaadja az *elso* mutató által meghatározott binárisfa elemeinek összegét: **int szumma_bin(struct tag * elso)**. Feltehetjük, hogy nem üres a fa.

2. Adott a következő típusdefiníció:

```
struct elem1{int adat; struct elem1 * kov;};
```

   Írjon függvényt, mely visszaadja a *gy* gyökerű láncolt lista elemeinek a számát: **int darab(struct elem1 * gy)**. Feltehetjük, hogy nem üres a lista.

3. Implementálja C nyelven a következő algoritmust. Az egész számokat tartalmazó N elemű X és az M elemű Y és a Z tömb indexelése egytől kezdődik:

```
Algoritmus tombosalg(X, N, Y, M, Z, P, Q)
        i:=1
        j:=Q
        k:=0
        Amig  (i<=M valamint j<=N)
                Ha X[j]=Y[i] akkor
                        k:=k+1
                        Z[k]:=i
                Ha vege
                i:=i+1
        Amig vege
        P:=k
Algoritmus vege
```

4. Írjon függvényt, mely egy karakterekből álló N elemű X tömb elemeit kettesével megcserélgeti. Ha az elemek száma nem osztható kettővel, akkor az utolsó elem helyben marad:ABCDEFG⇒BADCFEG **void csere(char * X, int N)**. Feltehetjük, hogy legalább egy elem van a tömbben.

5. Adott a következő típusdefiníció, egy SOR típusú adatszerkezet elemeinek típusa:

```
struct elem2{char betu;struct elem2 * m1;};
```

Írjon függvényt, mely betesz egy elemet a SOR viselkedésű láncolt adatszerkezetbe, X a sor első elemének mutatója, és írni a sor végére szabad csak! Feltehetjük, hogy nem üres a sor. **void berak(struct elem2 * X, char ujbetu)**