A1. feladat

A `szamok.txt` szöveges fájl csupa számjegyet tartalmaz elválasztó jel nélkül. Írjon programot, mely összeszámolja és a standard kimenetre kiírja, hogy hány N-jegyű prímszám fedezhető fel a szöveges fájlban. Az N értéket a program a standard bemeneten kapja meg.

Írja meg a teljes programot. A prímtesztelésre külön függvényt írjon. Feltételezheti, hogy az N-jegyű számok elférnek az `unsigned` típusban. Feltételezheti, hogy a fájl legalább N számjegyet tartalmaz.

Naiv megoldás:

```c
#include <stdio.h>
#include <stdlib.h>

int isprime(unsigned N)
{
  unsigned n;
  for (n = 2; n*n <= N; ++n)
    if (N % n == 0)
      return 0;
  return 1;
}

int main(void)
{
  char *digits, i, N, n_primes = 0;
  unsigned num;

  FILE *fp = fopen("szamok.txt", "r");
  scanf("%d", &N);              /* méret beolvasása */
  digits = (char *)malloc(sizeof(char)*N);   /* foglalás */

  for (i = 0; i < N; ++i)    /* puffer feltöltése */
    fscanf(fp, "%c", digits+i);

  do {
    unsigned num = 0;      /* az aktuális szám */
    for (i = 0; i < N; ++i)    /* szám előállítása */
      num = 10 * num + digits[i] - '0';
    if (isprime(num))      /* prímteszt */
      n_primes++;
    for (i = 0; i < N-1; ++i) /* léptetés */
      digits[i] = digits[i+1];
  } while(fscanf(fp, "%c", digits+N-1) == 1);

  free(digits);
  fclose(fp);
  printf("%d", n_primes);
  return 0;
}
```

Hatékonyabb megoldás:

```c
#include <stdio.h>

int isprime(unsigned N)
{
  unsigned n;
  for (n = 2; n*n <= N; ++n)
    if (N % n == 0)
      return 0;
  return 1;
}

int main(void)
{
  int *digits, N, n_primes = 0, pos = 0, i;
  char c;
  unsigned num = 0; /* az aktuális N-jegyű szám */
  unsigned maxpow;  /* 10 az (n-1)-ediken */
  FILE *fp = fopen("szamok.txt", "r");
  scanf("%d", &N);
  digits = (int *)malloc(sizeof(int)*N);   /* foglalás */
```

```

  for (i = 0, maxpow = 1; i < N-1; ++i)
    maxpow *= 10;

  while (fscanf(fp, "%c", &c) == 1) {
    num = 10 * (num - maxpow * digits[pos]) + (c - '0');
    digits[pos] = c - '0';
    pos = (pos+1) % N;
    if (num >= maxpow && isprime(num))
      ++n_primes;
  }

  fclose(fp);
  printf("%d", n_primes);
  free(digits);
  return 0;
}
```

A2. feladat

Egy szöveg legfeljebb 20 karakter hosszú szavairól szeretnénk előfordulási statisztikát készíteni. A szavakat elöl strázsás egyirányú láncolt listában tároljuk, a szó mellé feljegyezve az előfordulások számát. Hogy hatékonyabb legyen a statisztikát készítő program, a listában a leggyakrabban előforduló szavak vannak legelöl, azonos darabszám esetén pedig a legutóbb olvasottak.

- Definiálja a `Szo` típust, mely egy listaelemet valósít meg!
- Készítse el azt a funkciót C nyelven, amely egy szó beolvasása után a szavak halmazát, illetve darabszámukat frissíti. A függvény a listát és a beolvasott szót mint sztringet kapja paraméterül. Ha a szó hosszabb, mint 20 betű, akkor nem kell foglalkoznia vele.

```c
#include <stdlib.h>
#include <string.h>

typedef struct Szo {
  char szo[20+1];
  int db;
  struct Szo *kov;
} Szo;

void berak(char *szo, Szo *fej)
{
  Szo *lemarado = fej, *uj;
  if (strlen(szo) > 20)
    return;
  /* megkeressük a beérkező szót előretartással */
  while (lemarado->kov != NULL && strcmp(lemarado->kov->szo, szo) != 0)
    lemarado = lemarado->kov;
  if (lemarado->kov == NULL) {  /* nincs meg */
    uj = (Szo *)malloc(sizeof(Szo));
    strcpy(uj->szo, szo);
    uj->db = 1;
  }
  else {  /* megvan */
    uj = lemarado->kov;   /* megfogjuk */
    uj->db++;
    lemarado->kov = uj->kov;  /* kiláncoljuk */
  }
  /* beillesztési pozíció megkeresése előretartással */
  lemarado = fej;
  while (lemarado->kov != NULL && lemarado->kov->db > uj->db)
    lemarado=lemarado->kov;
  uj->kov = lemarado->kov;  /* beláncoljuk */
  lemarado->kov = uj;
}
```

A3. feladat

Egy íjászversenyen minden íjász hármat lőhet. A lövések pozícióját a céltábla középpontjától mért vízszintes és függőleges koordináták formájában tartjuk nyilván. A versenyen használt céltáblát tíz számérték írja le, ezek a céltábla gyűrűinek vastagságai. A legelső gyűrű a legbelső, ez körlappá fajul, vastagsága a kör sugara, és tíz pontot ér. A következő gyűrűk egyre kevesebb pontot érnek, a legszélső (tizedik) az egypontos.
Definiáljon adatszerkezetet

- egy lövés leírására
- egy íjász leírására (egész rajtszám + három lövés tömbje)
- egy tábla leírására (egész azonosító + gyűrűk vastagságainak tömbje)

Írjon függvényt, mely paraméterként megkapja az íjászok tömbjét és az íjászok által használt közös céltáblát. A függvény adja vissza, hogy melyik rajtszámú íjász érte el a legtöbb pontot. Egy íjász pontszámát külön függvénnyel számolja ki!

```c
typedef struct {
  double x, y;
} Loves;

typedef struct {
  int rajtszam;
  Loves lovesek[3];
} Jatekos;

typedef struct {
  int sorszam;
  double szelessegek[10];
} Tabla;

int pontszam(Loves lovesek[], Tabla tabla)
{
  int l, pont = 0;
  for (l = 0; l < 3; ++l) {
    double R = 0.0;
    int r;
    for (r = 0; r < 10; ++r) {
      R += tabla.szelessegek[r];
      if (lovesek[l].x*lovesek[l].x+lovesek[l].y*lovesek[l].y < R*R) {
        pont += 10 - r;
        break;
      }
    }
  }
  return pont;
}

int legjobb_rajtszam(Jatekos jatekosok[], int n, Tabla tabla)
{
  int max_pont = pontszam(jatekosok[0].lovesek, tabla);
  int legjobb = jatekosok[0].rajtszam;
  int i;

  for (i = 1; i < n; ++i) {
    int p;
    if ((p = pontszam(jatekosok[i].lovesek, tabla)) > max_pont) {
      max_pont = p;
      legjobb = jatekosok[i].rajtszam;
    }
  }

  return legjobb;
}
```