Egyszerű gyakorló feladatok

1.) Írjon egy C programot, amely segít eldönteni, hogy egy diák sikeres volt-e egy vizsgán! A programnak a következő funkciókat kell megvalósítania:

- Kérje be a felhasználótól a maximális pontszámot (egész szám).
- Kérje be a felhasználótól az elért pontszámot (egész szám).
- Ellenőrizze, hogy az elért pontszám eléri-e legalább a maximális pontszám 60%-át.
- Kiírja a képernyőre a következő üzenetet:
  - Ha a feltétel teljesül (az elért pontszám legalább a maximális pontszám 60%-a): “Sikeres”
  - Ha a feltétel nem teljesül: “Sikertelen”

Tesztelje le néhány példával, hogy helyesen működik-e a vizsgaeredmények kiértékelése.

```
#include <stdio.h>
int main()
{
    printf("Kerem a maximum pontot:");
    int maxPont;
    scanf("%d", &maxPont);
    printf("Kerem az elert pontot:");
    int elertPont;
    scanf("%d", &elertPont);
    if(elertPont > maxPont * 0.6)
    {
        printf("Sikeres");
    }
    else
    {
        printf("Sikertelen");
    }
}
```

2. Írjon egy C programot, ami egy évszámról megállapítja, hogy szökőév-e?

Szökőévek számolása kapcsán a következő szabályokat alkalmazzuk:

- Oszthatóság 4-gyel: Az év szökőév, ha osztható 4-tel. Például: 2004, 2008, 2012.
- Kivétel a századok esetén: A szabálytól van egy kivétel: ha egy év század év (például 1800, 1900, 2000), akkor az csak akkor szökőév, ha osztható 400-zal. Tehát a 1900 nem szökőév, mivel bár osztható 4-tel, de század, és nem osztható 400-zal.

```
#include <stdio.h>

int main() {
```

```
    printf("Kerem az evet: ");

    int ev;
    scanf("%d", &ev);

    // Szökőév ellenőrzése
    if ((ev % 4 == 0 && ev % 100 != 0) || (ev % 400 == 0))
    {
        printf("%d egy szokoev.\n", ev);
    }
    else
    {
        printf("%d nem szokoev.\n", ev);
    }

    return 0;
}
```

3. Írjon programot, ami eldönti, hogy a felhasználó által bevitt három szakasz hossza alapján, a szakaszok alkothatnak-e háromszöget? Akkor szerkeszthető háromszög, ha bármely két szakasz hossza nagyobb mint a harmadik.

```
#include <stdio.h>

int main()
{
    // Bemenet: három szakasz hossza
    double a, b, c;

    printf("Kerem az elso szakasz hosszat: ");
    scanf("%lf", &a);

    printf("Kerem a masodik szakasz hosszat: ");
    scanf("%lf", &b);

    printf("Kerem a harmadik szakasz hosszat: ");
    scanf("%lf", &c);

    // Háromszög szerkeszthetőségének vizsgálata
    if (a + b > c && a + c > b && b + c > a) {
        printf("A haromszog szerkesztheto.\n");
    } else {
        printf("A haromszog nem szerkesztheto.\n");
    }

    return 0;
}
```

4. Írjunk C programot, amely egész számokat olvas be a billentyűzetről mindaddig, amíg 0-t nem gépelünk, és minden beolvasott számról eldönti, hogy páros-e vagy páratlan

```
#include <stdio.h>

int main() {
    int szam;

    // Olvassuk be az első számot
    printf("Kerem a szamot (0 a kilepeshez): ");
    scanf("%d", &szam);

    while (szam != 0) {
        // Eldöntjük, hogy páros vagy páratlan
        if (szam % 2 == 0) {
            printf("%d paros.\n", szam);
        } else {
            printf("%d paratlan.\n", szam);
        }

        // Olvassuk be a következő számot
        printf("Kerem a kovetkezo szamot (0 a kilepeshez): ");
        scanf("%d", &szam);
    }

    printf("Program kilepett, mert 0-t irtal be.\n");

    return 0;
}
```

5. Írjunk másodfokú egyenlet valós gyökeit megoldó programot. Bemenetkét kérjük be a három együtthatót.

```
#include <stdio.h>
#include <math.h>

int main() {
    double a, b, c;
    double discriminant, x1, x2;

    // Bemenet: a, b és c beolvasása
    printf("Kerem a masodfoku egyenlet egyutthatoit:\n");
    printf("a: ");
    scanf("%lf", &a);
    printf("b: ");
```

```
    scanf("%lf", &b);
    printf("c: ");
    scanf("%lf", &c);

    // Számláló (discriminant) kiszámítása
    discriminant = b * b - 4 * a * c;

    // Discriminant ellenőrzése
    if (discriminant > 0) {
        // Két valós gyök
        x1 = (-b + sqrt(discriminant)) / (2 * a);
        x2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("A masodfoku egyenletnek ket valos gyoke van:\n");
        printf("x1 = %lf\n", x1);
        printf("x2 = %lf\n", x2);
    } else if (discriminant == 0) {
        // Egy valós gyök
        x1 = -b / (2 * a);
        printf("A masodfoku egyenletnek egy valos gyoke van:\n");
        printf("x1 = %lf\n", x1);
    } else {
        // Nincsenek valós gyökök
        printf("Nincsenek valós gyokok");
    }

    return 0;
}
```

6. Írjunk c programot ami bekér a felhasználótól egy természetes számot, majd eldönti, hogy prímszám-e vagy sem. A prímszám ellenőrzésére egy egyszerű függvényt használ, ami a szám négyzetgyökéig megy és ellenőrzi, hogy osztható-e bármely számmal a [2, négyzetgyök(szám)] intervallumban .

```
#include <stdio.h>
#include <math.h>

// Függvény a prímszám ellenőrzésére
int isPrime(int number) {
    if (number < 2) {
        return 0;  // Nem prímszám, mert kisebb, mint 2
    }

    int i;
    int sqrtNumber = sqrt(number);

    for (i = 2; i <= sqrtNumber; ++i) {
```

```
        if (number % i == 0) {
            return 0;  // Nem prímszám, mert osztható más számmal is
        }
    }

    return 1;  // Prímszám
}

int main() {
    int szam;

    // Olvassuk be a számot
    printf("Kerem a szamot: ");
    scanf("%d", &szam);

    // Ellenőrizzük, hogy a szám prímszám-e
    if (isPrime(szam)) {
        printf("%d primszam.\n", szam);
    } else {
        printf("%d nem primszam.\n", szam);
    }

    return 0;
}
```

7. Kérjük be karaktereket az 'enter' megnyomásáig és közben számoljuk meg hány kisbetűt írt a felhasználó. (az angol ABC szerint)

```
#include <stdio.h>
#include <conio.h>

int main() {
    printf("Kerem a karaktereket (amig entert nem nyomunk): ");

    char karakter;
    int kisbetuSzamlalo = 0;
    // Olvassuk be a karaktereket, amíg az angol ábécé betűit kapjuk
    while ((karakter = getch()) != '\r' &&
           ((karakter >= 'A' && karakter <= 'Z') || (karakter >= 'a' &&
karakter <= 'z'))) {
        // Ellenőrizzük, hogy a karakter kisbetű-e
        if (karakter >= 'a' && karakter <= 'z') {
            kisbetuSzamlalo++;
        }
        printf("%c", karakter);
    }

    // Kiírjuk, hány kisbetű volt
    printf("\nA bevitt szovegben %d kisbetű volt.\n", kisbetuSzamlalo);
```

```
    return 0;
}
```

8. Írjon C programot, amely az ötöslottó számsorsolását modellezi. A program véletlenszerűen generál öt különböző számot 1 és 90 között, majd kiírja ezeket a számokat: (figyelve, hogy különböző számokat húzzon

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

#define SZAMOK 5
#define LEGKISEBB_SZAM 1
#define LEGNAGYOBB_SZAM 90

int main() {
    int lottoSzamok[SZAMOK];
    int i, j;

    // Inicializáljuk a random generátort a jelenlegi idő alapján
    srand(time(NULL));

    // Generálunk öt különböző számot 1 és 90 között
    for (i = 0; i < SZAMOK_SZAMA; ++i) {
        int ujSzam;
        do {
            ujSzam = rand() % LEGNAGYOBB_SZAM + LEGKISEBB_SZAM;

            // Ellenőrizzük, hogy az újonnan generált szám már nem szerepel-
e a listában
            for (j = 0; j < i; ++j) {
                if (ujSzam == lottoSzamok[j]) {
                    ujSzam = 0; // Az ellenőrzés nem sikerült, generáljunk
új számot
                    break;
                }
            }
        } while (ujSzam == 0);

        // Hozzáadjuk az újonnan generált számot a listához
        lottoSzamok[i] = ujSzam;
    }

    // Kiírjuk a generált számokat
    printf("Az otoslotto szamok: ");
    for (i = 0; i < SZAMOK; ++i) {
        printf("%d ", lottoSzamok[i]);
```

```
    }
    printf("\n");

    return 0;
}
```

9. Írjon programot ami a felhasználótól bekért szövegben a betűk gyakoriságát megállapítja, és egy hisztogrammot készít a konzolon.

A programban a betűk gyakoriságát egy 26 elemű tömbben (gyakorisag) érdemes tárolni, ahol minden elem egy betű gyakoriságát jelenti. A program kérje be a felhasználótól a szöveget, majd a betűk gyakoriságát számolja ki és egy hisztogrammal írja ki a konzolra. A toupper() függvény segítségével minden betűt nagybetűvé alakíthat, hogy az ábrázolás ne függjön a betűk nagy- vagy kisbetűs formájától.

```
#include <stdio.h>
#include <ctype.h>

#define MAX_HISZTOGRAM_MERET 26

void hisztogramKiirasa(int gyakorisag[]) {
    printf("\nBetuk gyakorisaga hisztogrammal:\n");

    for (int i = 0; i < MAX_HISZTOGRAM_MERET; ++i) {
        char karakter = 'A' + i;
        printf("%c | ", karakter);

        for (int j = 0; j < gyakorisag[i]; ++j) {
            printf("*");
        }

        printf("\n");
    }
}

int main() {
    char szoveg[1000];
    int gyakorisag[MAX_HISZTOGRAM_MERET] = {0}; // minden elemet nulláz

    printf("Kerem a szoveget (legfeljebb 1000 karakter): ");
    fgets(szoveg, sizeof(szoveg), stdin);

    for (int i = 0; szoveg[i] != '\0'; ++i) {
        char karakter = toupper(szoveg[i]);

        if (isalpha(karakter)) {
            int index = karakter - 'A';
            gyakorisag[index]++;
        }
```

```
    }

    hisztogramKiirasa(gyakorisag);

    return 0;
}
```

10. Rajzoljunk egy kört a konzolra csillag karakterekből.

Ebben a játékos feladatban, legyen egy circle() függvény ami kiszámolja egy adott pont távolságát a kör középpontjától, majd összehasonlítja a sugárral. Ha a távolság közel van a sugárhoz, a függvény közelítőleg 0-t ad vissza.

```
#include <stdio.h>
#include <math.h>
const float centerX = 0.5;
const float centerY = 0.5;

const int screenSizeX = 50;
const int screenSizeY = 30;

float circle(float x, float y, float r)
{
    return (x - centerX) * (x - centerX) + (y - centerY) * (y - centerY) - r
* r;
}

int main() {
    for(int j = 0; j <= screenSizeY; j++)
    {
        for(int i = 0; i <= screenSizeX; i++)
        {
            float x = (float)i / screenSizeX;
            float y = (float)j / screenSizeY;
            if(fabs(circle(x, y, 0.3f)) < 0.01f)
            {
                printf("*");
            }
            else
            {
                printf(" ");
            }
        }
        printf("\n");
    }
    return 0;
}
```