5. labor kis ZH, 2024. május 16., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

max. 10p, megoldási idő: 15 perc

1. Adott az alábbi Ember osztály. Származtass belőle Hallgato osztályt, amely pluszként a megszerzett kreditek mennységét (egy egész szám) tárolja! Készíts konstruktort, amely a hallgató minden adatát (az Ember-hoz tartozókat is) inicializálja! Definiáld felül a print függvényt úgy, hogy az a hallgató minden adatát írja ki! Az Ember osztályt nem módosíthatod! (3p)

```
class Ember {
    int kor;
public:
    Ember(int age): kor{ age } {}
    void print()const { std::cout << "kor=" << kor; }
};

class Hallgato :public Ember { // 1p
    int kredit;
public:
    Hallgato(int age, int kredit): Ember(age), kredit{ kredit } {} // 1p
    void print()const {
        Ember::print(); // 1p
        std::cout << ", kredit: " << kredit;
    }
};
```

2. Adott a következő kódrészlet:

```
Ember e(32); Hallgato h(20, 60); kiir(e); kiir(h);
```

Készítsd el a kiir függvényt (1 db függvény), amely meghívja az átvett objektum print függvényét! Mit kell módosítani az Ember és/vagy Hallgato osztályon, hogy a kiir függvény hívásakor mindig a helyes print függvény hívódjon meg? Írd le ezt is (az 1. feladatba is beleírhatod, de jelöld!)! (2p)

```
void kiir(const Ember& e) { // 1p
    e.print();
}


class Ember {
    int kor;
public:
    Ember(int age): kor{ age } {}
    virtual void print()const { std::cout << "kor=" << kor; } // 1p
};
```

3. Készíts generikus függvényt, amely egy tömböt kap paraméterként, és visszaadja a tömb legnagyobb elemének indexét! Készíts specializációt C sztringekre mutató pointerek tömbje esetére, amely a leghosszabb sztring indexét adja vissza! Írj kódrészletet, amelyben kipróbálod a függvényt double tömbre és C sztringpointer tömbre is! (A kipróbálásnál felhasználhatod a következő sztringpointer tömböt.) (6p)

```
const char* sptomb[4] = { "Dénes", "Aladár", "Cecil", "Béla" };
```

```cpp
template<typename T> // 1p
int legnagyobb(const T t[], int n) { // 1p
    int max = 0;
    for (int i = 1; i < n; i++) // alg: 1p
        if (t[i] > t[max])
            max = i;
    return max;
}

template<>
int legnagyobb(const char* const t[], int n) { // specializáció 1p
    int max = 0;
    for (int i = 1; i < n; i++) // alg: 1p
        if (strlen(t[i]) > strlen(t[max]))
            max = i;
    return max;
}
    double dt[6] = { 10, -8, 0, 99, -11, 3 };
    const char* sptomb[4] = { "Dénes", "Aladár", "Cecil", "Béla" };
    std::cout << legnagyobb (dt, 6) << std::endl; // ha mindkét példa jó: 1p
    std::cout << legnagyobb (sptomb, 4) << std::endl;
```