Eredménye:	Név:	NEPTUN kód:
	Gyakorlatvezető:	Piszkozatlapok: darab / Csoport: A

FONTOS! A feladatok megoldására **90 perc** áll rendelkezésre. A NEPTUN-kódját a piszkozatlapokon is tüntesse fel! A feladatok megoldásához kék színű tollat használhat, más segédeszköz nem megengedett. A teremben kamerás megfigyelés működik!

1. Mi lesz a következő program kimenete?
   (Ügyeljen a konstruktorhívásokra is!)

```
#include <stdio.h>

class A {
public:
    A() { Print(); }
    void A::Print() { printf("Class A\n"); }
};

class B : public A {
public:
    B() : A() { Print();  }
    void B::Print() { printf("Class B\n"); }
};

int main()
{
    B b;
    A &a = b;
    a.Print();
}
```

   A program kimenete:
   Class A
   Class B
   Class A

2. Készítse el az alábbi osztályban található metódusok implementációját.
   Az osztály feladata időmennyiségek tárolása óra, perc és másodperc formában. A definíciók az osztály törzsén kívülre kerüljenek! Ügyeljen arra, hogy a konstruktorok úgy inicializálják a mezőket, hogy a perc és másodperc értéke 60 alatt legyen.

```
class Time {
private:
    int Hours, Minutes, Seconds;
public:
    Time(int Hour, int Minute, int Seconds);
    Time(int Seconds);
    int operator==(const Time &T);
};

Time::Time(int Hour, int Minute, int Seconds) {
    Time::Seconds = Hour * 60 * 60 + Minute * 60 + Seconds;
    Time::Hours = Time::Seconds / (60 * 60);
    Time::Minutes = (Time::Seconds / 60) % 60;
    Time::Seconds = Time::Seconds % 60;
}

Time::Time(int Seconds) {
    Time::Hours = Seconds / (60 * 60);
    Time::Minutes = (Seconds / 60) % 60;
    Time::Seconds = Seconds % 60;
}

int Time::operator==(const Time &T) {
    return (Hours == T.Hours && Minutes == T.Minutes
            && Seconds == T.Seconds) ? 1 : 0;
}
```

3. Egészítse ki a `Time` osztályt az alábbi metódus deklarációjával és definíciójával úgy, hogy az összeadás eredménye valóban akét időmennyiség összege legyen! A metódus dobjon kivételt, ha az összeg eléri a 24 órát!

```
Time operator+(const Time &T1, const Time &T2);
```

Az osztály deklarációja a következő sorral egészül ki:

```
friend Time operator+(const Time &T1, const Time &T2);
```

A függvény definíciója (az osztály törzsén kívül) a következő:

```
Time operator+(const Time &T1, const Time &T2) {
    Time result = Time(T1.Hours + T2.Hours, T1.Minutes + T2.Minutes,
            T1.Seconds + T2.Seconds);
    if (result.Hours >= 24)
        throw "HIBA!";
    return result;
}
```

4. Készítse el az alábbi osztályban található metódusok implementációját.
   A `SumX` és `SumY` metódusok feladata a tábla soronkénti és oszloponkénti összegének előállítása. A definíciók az osztály törzsén kívülre kerüljenek!

```
template<int N, int M> class Table {
public:
    int t[N][M];
    Table() {
        for (int i = 0; i <= N; i++) {
            for (int j = 0; j <= M; j++) {
                t[i][j] = (i + 1)*(j + 1);
            }
        }
    }
    Table<N, 1> SumX();
    Table<1, M> SumY();
};

template<int N, int M> Table<N, 1> Table<N, M>::SumX() {
    Table<N, 1> r;
    for (int i = 0; i <= N; i++) {
        r.t[i][0] = 0;
        for (int j = 0; j <= M; j++)
            r.t[i][0] += t[i][j];
    }
    return r;
}

template<int N, int M> Table<1, M> Table<N, M>::SumY() {
    Table<1, M> r;
    for (int j = 0; j <= M; j++) {
        r.t[0][j] = 0;
        for (int i = 0; i <= N; i++)
            r.t[0][j] += t[i][j];
    }
    return r;
}
```

5. Röviden ismertesse, hogy mikor érdemes `const` kulcsszót használni a metódusdeklarációk végén.

   A `const` kulcsszó egy tagfüggvény deklarációjának végén azt jelzi, hogy a függvény nem változtatja meg az aktuális példány adattagjainak értékét. Az ilyen tagfüggvények konstans objektumok esetén is hívhatóak.

6. Röviden ismertesse, hogy mit jelent a `protected:` cimke osztálydeklarációban.

   A `protected:` címkét követően deklarált erőforrások hatásköre a `private:` címkével deklarálttakénál annyival bővebb, hogy a tekintett osztályból származtatott osztályokra is kiterjed.