Vytvořte **šablonu** třídy **Matice** představující **2D matici**. Jediným parametrem šablony je obecný typ dat (typ hodnot v matici). Třída bude obsahovat 3 atributy: T** pro uchování prvků matice, počet řádků, počet sloupců.

Matici implementujte v souboru matice.h, metody definujte VNĚ (mimo) deklaraci třídy!

Dále realizujte:

- **`konstruktor(int radky, int sloupce)`** – inicializuje matici hodnotami 0
- **`kopírovací konstruktor(const Matice<T>& m)`** – vytvoří matici jako kopii jiné (vytvoří hlubokou kopii)
- **`destruktor`** – dealokuje matici
- metoda **`void Nastav(int radek, int sloupek, T hodnota)`** – nastaví vybraný prvek v matici na specifikovanou hodnotu

- metoda **`void NastavZ(T* pole)`** – nastaví všechny buňky podle vybraného jednorozměrného pole (předpokládejte, že rozměr pole je správný; hodnoty v poli jsou uloženy postupně po řádcích)

  `pole = {1,2,3,4,5,6}, radku = 2, sloupcu = 3` ➔

1	2	3
4	5	6

- pro přístup k buňkám realizujte metody (přístup na neplatný index vyvolá výjimku):
  `T& Dej(int radek, int sloupec)`
  `const T& Dej(int radek, int sloupec) const`

- šablonovou metodu (vnořená šablona s obecným typem R)
  `Matice<R> Pretypuj() const` – metoda vytvoří novou matici požadovaného typu o stejných rozměrech jako původní matice a převede všechny hodnoty na cílový typ do nové matice

- dále realizujte základní maticové operace (operace mezi maticemi nebo mezi maticí a skalárem; ošetřete výjimkami neplatné velikosti matic):
  `Matice Transpozice() const`
  `Matice Soucin(const Matice& m) const`
  `Matice Soucin(T skalar) const`
  `Matice Soucet(const Matice& m) const`
  `Matice Soucet(T skalar) const`
  Operace vždy vytvářejí novou matici a nemění aktuální objekt.

- pro porovnání shody dvou matic vytvořte metodu
  `bool JeShodna(const Matice& m) const`

- pro výpis matice na terminál
  `void Vypis() const`

Otestujte správnou funkčnost matice ukázkovým kódem použití třídy (vizte druhou stranu zadání).

Neplatná volání ošetřete výjimkami (neplatný index u Nastav/Dej, neplatný rozměr matice u Soucin/Soucet).

Vypracovanou šablonu třídy Matice odevzdejte na STAG.

```
int main()
{
	Matice<int> m{ 3,3 };
	int jednodpole[] = { 0,1,2,3,4,5,6,7,8 };
	m.NastavZ(jednodpole);

	Matice<int> mt = m.Transpozice();
	Matice<int> mmt = m.Soucin(mt);

	Matice<double> mmt_d = mmt.Pretypuj<double>();

	Matice<double> n_d{ 3,3 };
	double jednodpole_d[] = { 4.5,5,0,2,-0.5,7,1.5,9,6 };
	n_d.NastavZ(jednodpole_d);

	Matice<double> mmtn_d = mmt_d.Soucin(n_d);

	Matice<int> r = mmtn_d.Pretypuj<int>();

	Matice<int> t{ 3,3 };
	int tpole[] = { 85,225,236,292,819,866,499,1413,1496 };
	t.NastavZ(tpole);

	std::cout << "r==t ? " << (r.JeShodna(t) ? "true" : "false") << std::endl;
	return 0;
}
```

Poznámky:

- uvedený kód nevyžaduje implementaci operátoru= pro konstrukci objektu, vyžadován je pouze kopírovací konstruktor vytvářející hlubokou kopii
- definované rozhraní je kompletní pro realizaci požadovaných funkcionalit není potřeba žádné další metody nebo atributu či parametru
- při správné implementaci musí vyjít na konci „true“; ukázkový kód netestuje všechny metody – součty a skalární součin otestujte sami!
- Složitost operací (typický počet cyklů v metodě pro implementaci):
    - Konstruktor(int, int) – 1 for (alokace polí)
    - Kopírovací konstruktor – 2 for (alokace + kopírování)
    - Destruktor – 1 for (dealokace polí)

    - NastavZ – 2 for
    - Pretypuj – 2 for
    - Transpozice – 2 for
    - Soucin (maticový) – 3 for
    - Soucin (skalár) – 2 for
    - Soucet (maticový, skalární) – 2 for
    - JeShodna – 2 for
    - Vypis – 2 for

    - Ostatní metody – nevyžadují cykly