test_1_lex_err

```
/* Program 1: Vypocet faktorialu (iterativne) */
int main()
{
	int 1a; [1; chybny lexem; 1]
	int vysl;
	cout << "Zadejte cislo pro vypocet faktorialu: ";
	cin >> a;
	$ [2; chybny lexem; 1]
	if (a < 0) // cin nacitani zaporneho cisla nemusi podporovat
	{
		cout << "Faktorial nelze spocitat!\n";
	}
	else
	{
		vysl ?= 1; [3; chybny lexem; 1]
		for (int foo; a > 0; a = a - 1)
		{
			vysl = vysl * a;
		}
		cout << "Vysledek je: " << vysl << "\n";
	] [4; chybny lexem; 1]
	return 0;
}
```

test_2_lex_err

```
/* Program 2: Vypocet faktorialu (rekurzivne) */
int factorial(int 22_n) // Definice funkce pro vypocet hodnoty faktorialu [1; chybny lexem; 1]
{
	int temp_result;
	auto decremented_n = n % 1; [2; chybny lexem; 1]
	if (n < 2) {
		return 1;
	} else # [3; chybny lexem; 1]
		temp_result = factorial(decremented_n);
		return n * temp_result;
	}
}

int main()
{
	int a; int vysl;
	cout << "Zadejte cislo pro vypocet faktorialu: ";
	cin >> a;
	if (a < 0) {
		cout << "Faktorial nelze spocitat!\n";
	} else {
		vysl = factorial(a);
		auto neg = 0 - vysl;
		cout << "Vysledek: " << vysl << " (zaporny: " << neg << ")\n";
	}

	return 0;
}
```

test_3_lex_err

```
/* Program 3: Prace s retezci a vestavenymi funkcemi */
int main()
{
	string str1;
	{ // vnoreny blok s lokalni promennou str2 a pristupem k str1
		int x;
		str1 ! "Toto je nejaky text"; [1; chybny lexem; 1]
		string str2;
		str2 = concat(str1, ", ktery jeste trochu obohatime");
		x = find(str2, "text");
		cout | "Pozice retezce \"text\" v retezci str2: " [2; chybny lexem; 1]
			<< x << "\n";
		cout << "Zadejte nejakou posloupnost vsech malych pismen a-h, "
			<< "pricemz se pismena nesmeji v posloupnosti opakovat:";
	}
	cin >> str1;
	str1 = sort(str1);
	if (str1 != "abcdefgh")
	{
		for (auto s = str1; s != "abcdefgh"; s = s)
		{
			cout << "Spatne zadana posloupnost, zkuste znovu:";
			cin >> str1;
			s = sort(str1);
		}
	}
	else ~ [3; chybny lexem; 1]
	return 0;
}
```

test_1_syn_err

```
/* Program 1: Vypocet faktorialu (iterativne) */
int main()
{
	int a; [1; chybna syntaxe; 2]
	int vysl;
	cout << "Zadejte cislo pro vypocet faktorialu: ";
	cin >> a; [2; chybna syntaxe; 2]

	if (a < 0) // cin nacitani zaporneho cisla nemusi podporovat [3; chybna syntaxe; 2]
	{ [4; chybna syntaxe; 2]
		cout << "Faktorial nelze spocitat!\n";
	}
	else
	{
		vysl = 1;
		for (int foo; a > 0; a = a - 1) [5; chybna syntaxe; 2]
		{
			vysl = vysl * a;
		}
		cout << "Vysledek je: " << vysl << "\n"; [6; chybna syntaxe; 2]
	}
	return 0;
} [7; chybna syntaxe; 2]
```

test_2_syn_err

```
/* Program 2: Vypocet faktorialu (rekurzivne) */
int factorial(int n) // Definice funkce pro vypocet hodnoty faktorialu [1; chybna syntaxe; 2]
{
	int temp_result;
	auto decremented_n = n - 1; [2; chybna syntaxe; 2]
	if (n < 2) {
		return 1;
	} else {
		temp_result = factorial(decremented_n);
		return n * temp_result; [3; chybna syntaxe; 2]
	}
}

int main()
{
	int a, int vysl; [4; chybna syntaxe; 2]
	cout << "Zadejte cislo pro vypocet faktorialu: "; [5; chybna syntaxe; 2]
	cin >> a;
	if (a < 0) { [6; chybna syntaxe; 2]
		cout << "Faktorial nelze spocitat!\n"; [7; chybna syntaxe; 2]
	} else { [8; chybna syntaxe; 2]
		vysl = factorial(a);
		auto neg = 0 - vysl;
		cout << "Vysledek: " << vysl << " (zaporny: " << neg << ")\n"; [9; chybna syntaxe; 2]
	}

	return 0;
}
```

test_3_syn_err

```
/* Program 3: Prace s retezci a vestavenymi funkcemi */
int main() [1; chybna syntaxe; 2]
{
	string str1;
	{ // vnoreny blok s lokalni promennou str2 a pristupem k str1
		int x;
		str1 = "Toto je nejaky text"; [2; chybna syntaxe; 2]
		string str2;
		str2 = concat(str1, ", ktery jeste trochu obohatime");
		x = find(str2, "text"); [3; chybna syntaxe; 2]
		cout << "Pozice retezce \"text\" v retezci str2: "
			<< x << "\n";
		cout << "Zadejte nejakou posloupnost vsech malych pismen a-h, "
			<< "pricemz se pismena nesmeji v posloupnosti opakovat:";
	}
	cin >> str1;
	str1 = sort(str1); [4; chybna syntaxe; 2]
	if (str1 != "abcdefgh")
	{
		for (auto s = str1; s != "abcdefgh"; s = s)
		{
			cout << "Spatne zadana posloupnost, zkuste znovu:";
			cin >> str1;
			s = sort(str1);
		}
	}
	else {} [4; chybna syntaxe; 2]
	return 0;
}
```

test_1_sem_err

```
/* Program 1: Vypocet faktorialu (iterativne) */
int main()
{
	int a; [1; nedefinovana promenna; 3]
	int vysl;
	cout << "Zadejte cislo pro vypocet faktorialu: ";
	cin >> a;
	int a; [2; redefinice promenne; 3]
	if (a < 0) // cin nacitani zaporneho cisla nemusi podporovat
	{
		cout << "Faktorial nelze spocitat!\n";
	}
	else
	{
		vysl = "b"; [4; typova kompatabilita; 4]
		for (int foo; a > 0; a = a - 1)
		{
			vysl = vysl2 * a; [3; nedefinovana promenna; 3]
		}
		cout << "Vysledek je: " << vysl << "\n";
	}
	return 0;
}
```

test_2_sem_err

```
/* Program 2: Vypocet faktorialu (rekurzivne) */
int factorial(int n) // Definice funkce pro vypocet hodnoty faktorialu
{
	int temp_result;
	auto decremented_n = n - 1;
	if (n < 2) {
		return 1;
	} else {
		temp_result = factorial2(decremented_n); [1; nedefinovana funkce; 3]
		return n * result; [2; nedefinovana promenna; 3]
	}
}

int main()
{
	int a; int vysl;
	cout << "Zadejte cislo pro vypocet faktorialu: ";
	cin >> a;
	if (a < 0) {
		coutt << "Faktorial nelze spocitat!\n"; [4; nedefinovana promenna; 3]
	} else {
		vysl = factorial(a, 2); [3; chybny pocet parametru; 4]
		auto neg = 0 - vysl + "tmp"; [5; chyba při odvozovani typu promenne; 5]
		cout << "Vysledek: " << vysl << " (zaporny: " << neg << ")\n";
	}

	return 0;
}
```

test_3_sem_err

```
/* Program 3: Prace s retezci a vestavenymi funkcemi */
int main()
{
	int str1; [1; typova kompatabilita; 4]
	{ // vnoreny blok s lokalni promennou str2 a pristupem k str1
		int x;
		str1 = "Toto je nejaky text";
		string str2;
		string x; [2; redefinice promenne; 3]
		str2 = concat(str1, ", ktery jeste trochu obohatime", x); [3; chybny počet parametru; 4]
		x = find(1000, "text"); [4; chybny typ parametru; 4]
		cout << "Pozice retezce \"text\" v retezci str2: "
			<< x << "\n";
		cout << "Zadejte nejakou posloupnost vsech malych pismen a-h, "
			<< "pricemz se pismena nesmeji v posloupnosti opakovat:";
	}
	cin >> str1;
	str1 = bubblesort(str1); [5; nedefinovana funkce; 3]
	if (str1 != "abcdefgh")
	{
		for (auto s = str1; s != "abcdefgh"; s = s)
		{
			cout << "Spatne zadana posloupnost, zkuste znovu:";
			cin >> str1;
			s = sort(str1);
		}
	}
	else {}
	return 0;
}
```