Kolejność zapisu reguł w języku PROLOG:
Ma znaczenie tylko przy regułach rekurencyjnych

W celu zadeklarowania funkcji rekurencyjnej w języku F# należy zastosować instrukcję:
let rec

Operator równości wartości numerycznych w Prologu jest definiowany jako: **=:=**

Operator koniunkcji jest w Prologu zdefiniowany jako: **, (przecinek)**

Który zestaw reguł języka Prolog pozwoli sprawdzić czy element jest na liście?:
(chyba) zawiera(Elem, [Elem| _]).
zawiera(Elem, [_ | Reszta]) :- zawiera(Elem, Reszta).

Które z wyrażeń języka C# tworzy wyrażenie lambda?: **(chyba) x => x + 1;**

Metoda Func<double, double> F(int f) {} jest: **funkcją wyższych rzędów**

Która metoda języka C# jest funkcją wyższych rzędów:
Func<double, double> F(int f) {}

Która metoda języka C# jest poprawnie zdefiniowana:

```
async void Metoda() {
  int x = await Metoda2();
  Console.WriteLine(„KOMUNIKAT”);
}
```

Ze wskaźników w języku C# można korzystać tylko w funkcji oznaczonej słówkiem kluczowym:
unsafe

Typ String w języku C# jest typem: **niemodyfikowalnym**

W interfejsach inwariantnych zmienna typu może pojawić się tylko jako:

Interfejs kontrawariantny jest definiowany instrukcją: **interface IInterface<in T> {}**

Deklaracja języka C# interface IInterfejs<in T> {}; :
Ustanawia nowy interfejs kontrawariantny
Inwariancja było by interface IInterfejs<T>, Kowariancja było by interface IInterfejs<out T>

W języku C# inwariantne zmienne typu mogą być wykorzystane: **(chyba) tylko do określ typu parametrów wejściowych funkcji**

Która deklaracja klasy jest poprawna: **class MojaKlasa<T> where T : new() { }**

W języku C# instrukcja g(12, 23) + f(12, 23) == f(12, 23) + g(12, 23):
zwróci true lub false w zależności od postaci funkcji f i g

W celu zaimportowania metod statycznych z klasy Przykład tak aby nie trzeba było podawać nazwy klasy przy ich wywołaniach, w języku C# należy użyć instrukcji: **using static Przykład;**

Wartościami niemodyfikowalnymi nazywamy takie wartości złożone (C#), które:
Mogą być inicjowane tylko w konstruktorze

Która deklaracja interfejsu w języku C# jest poprawna:
interface MojInterfejs<in T> { void Funkcja(T x); }

Która instrukcja języka C# poprawnie deklaruje metodę rozszerzającą:
static class Klasa { public static void F(this double x) {} }

Monadą nazywamy typ, który będzie wyposażony w funkcje: **bind i return**

Monady Either można użyć w programach napisanych w stylu funkcyjnym do: **obsługi błędów**

Operacja mapowania jest w LINQ zdefiniowana jako funkcja: **select**

Operacja sortowania w LINQ jest zdefiniowana jako: **OrderBy**

Jeżeli chcesz w języku C# zdefiniować kilka zadań asynchronicznych i wymagasz, aby program wykonywał się dalej po zakończeniu pierwszej z nich wykorzystasz metodę:
Task.WaitAny

Po wykonaniu programu Task t = new Task(()=> { Console.Write(„A”);}); t.Start(); t.Wait(); Console.Write(„B”); t.ContinueWith(c=> { Console.Write(„C”); });
na ekranie pojawi się: **ABC**

Funkcje, które nie mają efektów ubocznych, a ich zawartość zależy tylko od parametrów nazywamy: **czystymi**

W rekurencji ogonowej: **wywołanie rekurencyjne musi być ostatnim wywołaniem funkcji**

Który z poniższych elementów nie jest obowiązkowy dla zmiennych: **wartość**

Instrukcja using static Przykład służy do: **Zaimportowania do pliku wszystkich elementów statycznych klasy Przykład**

Zmienną, której typ jest wywnioskowany przez kompilator w języku C# tworzymy za pomocą słowa kluczowego: **var**

Słowo kluczowe fixed pozwala: **ustawić zmienną w stałym miejscu pamięci**

Odpowiednikiem funkcji wyższych rzędów w programowaniu obiektowym jest wzorzec:
strategia

Niech dana będzie lista L. Operacja mapowania zwróci nową listę, która:
Ma tę samą długość co lista L

Operator >> w języku F# realizuje operację: **złożenia funkcji**

Złożenie funkcji w języku F# jest realizowane poprzez operator: >>

Która z poniższych operacji jest niedozwolona w języku F#: **niejawnej konwersji**

Sumowanie wszystkich elementów na liście jest przykładem operacji: **agregacji**

Który fragment kodu jest poprawny:
let rec fun x = if x <=1 then 1
else x + fun (x-1);;

Instrukcja języka F#:
type Operator =
    | Plus
    | Minus
Tworzy: **unię z dyskryminatorem**

Operator L1@L2: **tworzy nową listę bez modyfikacji istniejących**

Listy w języku F# są strukturą: **żadne z powyższych**

Dopasowywanie wzorców języku F# jest realizowane poprzez instrukcje: **match … with**

Które wyrażenie języka F# stworzy poprawnie listę 3-elementową: **[1; 2; 3];;**