!!!!!!!!F!!!

```
// Learn more about F# at http://fsharp.org

open System
//pole kola
let poleK r =
    Math.PI*pown r 2

let wynik = poleK 2.0
Console.WriteLine(wynik)


// fibonaci rekurencyjnie
let rec fibonacci n =
    let rec fib a b n =
        if n > 0 then
            fib b (a+b) (n-1)
        else
            b
    if n = 1 then
        1
    elif n = 2 then
        1
    else
        fib 1 1 (n-2)

let wyn = fibonacci 4

Console.WriteLine(wyn)

// listy laczone
let lista1 = [1;2;3;4];
let lista2 = [2;3;4;5];

let vecadd  = List.map2 (+)

let result  = vecadd lista1 lista2
printfn "%A" result
System.Console.WriteLine( result)

//zadania z listy f sharp xD

//F#: Napisz funkcję, która na podstawie określonej liczby
//rzutów trzema kostkami obliczy prawdopodobieństwo
//wyrzucenia kombinacji 1,2,3. Liczbę powtórzeń podaj jako parametr funkcji

let prawdopodobienstwo n:float =
    if n >=3.0 then
     (1.0/6.0)*(1.0/6.0)*(1.0/6.0)*(n-2.0)
    else 0.0
let wynikchuj = prawdopodobienstwo 3.0
Console.WriteLine( wynikchuj)

//F#: Zdefiniuj nowy typ Lista reprezentujący listę łączoną mogącą
//przechowywać wartości dowolnego typu. Następnie napisz funkcję,
//która będzie przyjmowała dwie takie listy liczb całkowitych i
//obliczała sumę odpowiadających sobie liczb. Przykładowo dla list
//(1,2,3,4); (2,3,4,5) funkcja powinna zwrócić listę (3,5,7,9).
//Napisz również kod, który demonstruje działanie przygotowanej funkcji.

let lista = [1;2;3;4]
let lista2x = [2;3;4;5]
let listLaczona = List.map2 (+)
let xD = listLaczona lista lista2x
printfn "%A" xD
//napisz funkcje ktora obliczy wartisc waruancji bez powtorzen zgodnie ze wzorem n!/(n-k)!
let rec silnia n =
    let rec sil x m =
        if m > 0 then
            sil (x*m) (m-1)
        else
            x
    sil 1 n
```

```
let war n k=
 if (n-k)=0 then
    0
 else
    let licznik=
        silnia n
    let mian =
        silnia(n-k)
    licznik/mian

let wynik4 = war 6 3

//Napisz funkcje ktora sprawidzi czy dany punkt znajduje sie wewnatrz prostokata Okresl odpowiednie
typy reprezentowania punktu oraz prostokata
type Punkt ={
x:int
y:int
    }
type Prostokat ={
x1:int
y1:int
x2:int
y2:int
x3:int
y3:int
x4:int
y4:int
    }

let MojPunkt = {x=3;y=2}
let MojProstokat = {x1=2;y1=2;x2=4;y2=2;x3=4;y3=4;x4=2;y4=4}

let Sprawdzenie (p:Punkt) (p2:Prostokat) :bool  =
    if (( p.x >= p2.x1 ) && (p.x<=p2.x3) && (p.y>=p2.y1) && (p.y<=p2.y3)) then
        true
    else
        false
let wynikspr = Sprawdzenie MojPunkt MojProstokat

//Napisz funkcję, która będzie obliczała pole i obwód określonego prostokąta. Do zamodelowania
parametru i wyniku tej funkcji
//wykorzystaj najlepszy wg siebie typ danych.

let poleObwod a b=
    let pole =
        a * b
    let obwod  =
        2*a+2*b
    let result = (pole ,obwod )
    result
let para = poleObwod 2 2
//Zdefiniuj nowy typ Lista reprezentujący listę łączoną mogącą przechowywać wartości dowolnego typu.
//Następnie napisz funkcję, zwracającą te elementy, które nie spełniają określonego warunku.
Zademonstruj jej działanie.

let testList = [1;2;3;4;5];

let DoNotMeetTheCondition list condition =
    let conditionNot = condition >> not
    let result = List.filter conditionNot list
    result


let result5 = DoNotMeetTheCondition testList (fun x -> x>3)
printfn "%A" result5

//Zdefiniuj nowy typ danych reprezentujący drzewo binarne.
//Następnie napisz program, który wyświetli elementy tego drzewa w kolejności postorder. Zademonstruj
jego działanie.

//Postorder: Najpierw wszystkie dzieci (lewe i prawe poddrzewo), potem rodzic (korzeń)
type drzewo =
| Puste
```

```
| Wezel of int * drzewo * drzewo


let rec pokazPostOrder  = function
| Puste -> ()
| Wezel(x, l, p) ->

    match l with
       | Wezel(_, _, _) -> pokazPostOrder l
       | Puste -> ()

    match p with
        | Wezel(_, _, _) -> pokazPostOrder p
        | Puste -> ()

    printfn "%d" x


let drzewo1:drzewo=
    Wezel(1,Puste,Puste)
let drzewo2:drzewo=
    Wezel(1,Puste,Puste)
let drzewo3:drzewo=
    Wezel(1,drzewo1,drzewo2)
pokazPostOrder drzewo1

//Napisz funkcję, która przyjmie parę liczb i określi, która z nich jest większa po podniesieniu do
kwadratu np.
//wywołanie wieksza (-5, 4) powinno zwrócić -5. Zastosuj dopasowanie do wzorca i wzorce warunkowe
let wieksze  a b =
    let a2 =
        a*a
    let b2 =
        b*b
    if (b2>a2) then
        b
    else
        a
let wieksza = wieksze -5 4

//Napisz program, który utworzy listę 100 dwuwymiarowych punktów losowych z przedziału od -20 do 20.
//Następnie napisz program, który wybierze te punkty, które znajdują się w promieniu r=5 od środka
układu współrzędnych.
//Wykorzystaj funkcje modułu List

let generuj n =
    let gll = new System.Random()
    List.init n (fun i -> (gll.Next(-21,21),gll.Next(-21,21)))
    |>List.map (fun (x,y)->(x,y,sqrt (((float) x)**2.0 + ((float)y)**2.0)))
    |>List.filter(fun (_,_,v) -> (v<5.0 ))
    |>List.iter (fun (x,y,v)->  printfn " (%d, %d,)" x y    )



generuj 100

//F#: Napisz funkcję rekurencyjną, która określi ile razy w danym łańcuchu znaków wystąpiła każda z
cyfr.
//Jeżeli jakaś cyfra nie występuje w tekście jej wartość w wynikach powinna zawierać 0

let countNumbers word =
    let rec countNumber (word:string) (char:char) i sum =
        if i=word.Length then
            (char, sum)
        else
            if (word.[i] = char) then
                countNumber word char (i+1) (sum+1)
            else
                countNumber word char (i+1) sum
    [for i in 48..57 -> countNumber word ((char)i) 0 0]


let result6 = countNumbers "Bartczuk2022"
```

```
printfn "%A" result6;

///F#: Napisz program, który utworzy listę 100 dwuwymiarowych punktów losowych z przedziału od -20 do
20.
//Następnie określ, odległość euklidesową każdego z tych punktów od początku układu współrzędnych.
Wykorzystaj funkcje modułu List.
let generuj2 n =
    let gll = new System.Random()
    List.init n (fun i -> (gll.Next(-21,21),gll.Next(-21,21)))
    |>List.map (fun (x,y)->(x,y,sqrt (((float) x)**2.0 + ((float)y)**2.0)))
    |>List.iter (fun (x,y,v)->  printfn " (%d, %d,)" x y     )

generuj2 100

//Zdefiniuj nowy typ Lista reprezentujący listę łączoną mogącą przechowywać wartości dowolnego typu.
//Następnie napisz funkcję, która obliczy ile wyrazów na tej liście zaczyna się na literę 'a'. Napisz
//również kod, który demonstruje działanie przygotowanej funkcji.
open System
let countAA (wordList:string list) =
     let sume1 = ref 0
     for word in wordList do
        if (word.[0] = 'A') then
            incr sume1
        else()

     sume1.Value

let listaA = ["Bartczuk2022";"Bilu2022";"Paradygmaty2022";"Achujztym";"AAAA"]
let result8 = countAA listaA
printfn "%A" result8;
    //Napisz program, który wczytując od użytkownika liczby z klawiatury zapamięta liczby parzyste.
///(Wprowadzanie zakończ, jeżeli użytkownik poda 0). Po zakończeniu wprowadzania danych program
//powinien wyświetlić je w odwrotnej kolejności (od ostatniej wprowadzonej do pierwszej).

let rec Wczytuj2 =
    let liczbaW ="xd"
    let listax2: int list=[]
    let rec Wcz (listax2: int list )=
        printf "podaj liczbe"
        let liczbaW = int(System.Console.ReadLine())
        if liczbaW = 0 then
          listax2 |>List.rev|>List.iter (fun x -> printfn "%d" x)

        else
            if (liczbaW) % 2 = 0 then

               let listax2  =List.append listax2 [liczbaW]
               Wcz listax2
            else
                Wcz listax2
    Wcz listax2

Wczytuj2


//F#: Napisz program, który utworzy listę 100 dwuwymiarowych punktów losowych z przedziału od -20 do
20.
//Następnie napisz program, który wybierze 10 punktów, które są najbliżej początku układu
współrzędnych.
//Wykorzystaj funkcje modułu List.
let generuj3 n =
    let gll = new System.Random()
    List.init n (fun i -> (gll.Next(-21,21),gll.Next(-21,21)))
    |>List.map (fun (x,y)->(x,y,sqrt (((float) x)**2.0 + ((float)y)**2.0)))
    |>List.sortBy (fun (_,_,v)->v)
    |>List.take 10
    |>List.iter (fun(x,y,v)->printfn ("(%d,%d)") x y)



generuj3 100
```

```
//Napisz program, który wczytując od użytkownika liczby z klawiatury, określi ile z nich było
parzystych.
//(Wprowadzanie zakończ, jeżeli użytkownik poda 0)


let rec Wczytuj =
    let licz = 0
    let liczbaW ="xd"
    let rec Wcz licz =
        printf "podaj liczbe"
        let liczbaW = System.Console.ReadLine()
        if liczbaW = "7" then
            licz
        else
            if Convert.ToInt32(liczbaW) % 2 = 0 then
                let licz = licz + 1
                Wcz licz
            else
                Wcz licz
    Wcz licz
let rez =Wczytuj
System.Console.WriteLine(rez)
```

!!C!!!1111

```
using System;
using System.Collections.Generic;
using System.Diagnostics.CodeAnalysis;
using System.Linq;

namespace cwiczeNaPAradygmaty
{
    //ZADANIE 1
    //C#: Zdefiniuj w C# klasę reprezentującą drzewo binarne.
    //Następnie napisz metodę, która policzy sumę wartości przechowywanych w węzłach,
    //spełniających warunek określony w funkcji przekazanej jako parametr. Zademonstruj
działanie swojego kodu.

    public class Node
    {
        public int key = 0;
        public Node Left = null;
        public Node Right = null;
        public Node(int x)
        {
            key = x;
        }
        public Node()
        {

        }

        public static Node Nowy(int wartosc)
        {
            Node node = new Node();
            node.key = wartosc;
            node.Left = null;
            node.Right = null;
            return node;
        }
        public static int addBT(Node root, Func<Node, bool> war)
        {
            if (root == null)
            {
                return 0;
            }
            if (war(root))
            {
                return (root.key + addBT(root.Left, war) + addBT(root.Right, war));
            }
            else { return (0 + addBT(root.Left, war) + addBT(root.Right, war)); }

        }
```

```
    }
    //zadanie2
    //C#: Napisz zapytanie LINQ, które dla dowolnej policzalnej kolekcji punktów
dwuwymiarowych określi ile z tych
    //punktów znajduje się powyżej funkcji sinus. Punkty generuj jako wartości
zmiennoprzecinkowe. Zakres wartości
    //dla współrzędnej X to -10.0 : 10.0. Zakres wartości dla współrzędnej Y określ
samodzielnie. Zademonstruj działanie
    //tego zapytania. Punkty mogą być reprezentowane w dowolny sposób.
    public class Punkt
    {
        public float x { get; set; }
        public float y { get; set; }
        public Punkt(float _x, float _y)
        {
            x = _x;
            y = _y;

        }

    }

    //zadanie3
    // //C#: Napisz funkcję obliczającą wartość silni w taki sposób, aby możliwe było jej
wywołanie w następujący sposób: 100.Silnia()
    public static class Ssilnia
    {

        public static int silnia(this int a)
        {
            int x = 1;
            for (int i = 1; i < a + 1; i++)
            {
                x = x * i;

            }
            return x;
        }


    }

    //zadanie4 i 5
    //2. C#: Stwórz generyczną klasę Lista implementującą listę łączoną i przechowującą
wartość dowolnego typu. Stwórz ją w taki sposób,
    //aby każdy węzeł był elementem tylko do odczytu. Następnie napisz w tej klasie metodę,
która policzy ile wartości dodatnich jest na tej
    //liście. Napisz również kod, który demonstruje jej działanie.
    public class MojaLista<T> {
    private List<T> lista= new List<T>();

        public List<T> Lista { get => lista; private set => lista = value; }

        public void Dodaj(T x) {

            Lista.Add(x);
        }
        public void Wyswietl() {
            Console.WriteLine("Elementy listy");
            for (int i = 0; i < Lista.Count; i++)
            {
                Console.WriteLine(Lista[i]);
            }
        }
        public void Policz()
        {
            int a = 0;
            for (int i = 0; i < Lista.Count; i++)
            {
                if (float.Parse(Lista[i].ToString())>0.0)
                {
                    a++;
                }
            }
            Console.WriteLine("liczb dodatnich jest: "+a);
        }
```

```
        public void ZnajdzNajmniejszy()
        {
            float a = float.Parse(Lista[1].ToString());
            for (int i = 0; i < Lista.Count; i++)
            {
                if (float.Parse(Lista[i].ToString()) < a)
                {
                    a= float.Parse(Lista[i].ToString());
                }
            }
            Console.WriteLine("liczb najmniejsza : " + a);
        }
    }
    //C#: Stwórz klasę Samochod zawierającą pola Marka, Model, PojemnoscSilnika. Zaimplementuj
w niej interfejs IComparable,
    //który pozwoli sortować instancje tej klasy wg. właściwości PojemnosciSilnika w
kolejności malejącej.
    //zadanie6
    public class Samochod:IComparable<Samochod>    {
        public string Marka { get; set; }
        public string Modell { get; set; }
        public int PojemnoscSilnika { get; set; }
        public Samochod(string _marka, string _model,int _pojemnosc)
        {
            Marka = _marka;
            Modell = _model;
            PojemnoscSilnika = _pojemnosc;


        }

        public int CompareTo([AllowNull] Samochod other)
        {
            {

                if (this.PojemnoscSilnika < other.PojemnoscSilnika)
                {
                    return 1;
                }
                else if (this.PojemnoscSilnika > other.PojemnoscSilnika)
                {
                    return -1;
                }
                else
                {
                    return 0;
                }
            }
        }
    }
    //zadanie 9
    public class Ksiazka {
        public string Tytul { get; set; }
        public int RokWydania { get; set; }
        public int Gatunek { get; set; }
        public float Cena { get; set; }


    }
    public class Ksiazka2
    {
        public string Tytul { get; set; }
        public int RokWydania { get; set; }
        public int Autor { get; set; }
        public float Cena { get; set; }


    }
    public class Autor
    { public int Id { get; set; }
        public string Imie { get; set; }
        public string Nazwisko { get; set; }



    }
```

```
    public class Gatunek {
        public int Id { get; set; }
        public string Nazwa { get; set; }


    }
    //zadanie13
    //C#: Napisz funkcję generującą leniwą kolekcję punktów w przestrzeni trójwymiarowej.
Każda składowa powinna być losową
    //liczbą całkowitą z przedziału -20 do 20. Punkty możesz reprezentować w dowolny sposób
np. jako tablicę lub obiekty klasy
    //Tuple. Następnie napisz funkcję, która pozwoli przerwać generowanie tych wartości, gdy
pojawi się punkt, którego trzecia
    //składowa będzie ujemna oraz funkcję zwracającą punkt, którego poszczególne składowe będą
sumą składowych wygenerowanych punktów.
    //Zademonstruj ich działanie.//zz13
    public class Punkt3d {
        public int X { get; set; }
        public int Y { get; set; }
        public int Z { get; set; }
        public Punkt3d(int x, int y ,int z)
        {
            X = x;
            Y = y;
            Z = z;
        }
        public static IEnumerable<Punkt3d> Losuj (){

            Random generator = new Random();
            Punkt3d p;

            while (true) { p = new Punkt3d(
               generator.Next(-20, 20), generator.Next(-20, 20), generator.Next(-20, 20)
                );
                if (p.Z < -18){ break; }

                yield return p;
            }




        }
    }
    internal class Program
        {
            static void Main(string[] args)
            {

            //ZADANIE 1
                //C#: Zdefiniuj w C# klasę reprezentującą drzewo binarne.
                //Następnie napisz metodę, która policzy sumę wartości przechowywanych w
węzłach,
                //spełniających warunek określony w funkcji przekazanej jako parametr.
Zademonstruj działanie swojego kodu.
                var node = new Node(1);
                node.Left = Node.Nowy(2);
                node.Left.Left = Node.Nowy(3);
                node.Left.Left.Right = Node.Nowy(4);

                Console.WriteLine(Node.addBT(node, (node) =>
                {
                    if (node.key > 1) { return true; }
                    else { return false; }
                }));
            //Stwórz klasy na podstawie przykładów: FOTA
            //zadanie9 i11

            var ksiazki = new[] {

           new Ksiazka {Tytul="Pan Tadeusz",RokWydania=1998,Gatunek=1,Cena=30 },
            new Ksiazka { Tytul = "Potop", RokWydania = 1991, Gatunek = 1, Cena = 40 },
            new Ksiazka { Tytul = "W pustyni i w puszczy", RokWydania = 1990, Gatunek = 2,
Cena = 30 },
            new Ksiazka { Tytul = "Lalka", RokWydania = 1990, Gatunek = 1, Cena = 50 },
            new Ksiazka { Tytul = "Programowanie funkcyjne w C#", RokWydania = 2019, Gatunek =
3, Cena =(float)71.20 },
```

```
            new Ksiazka { Tytul = "Programowanie funkcyjne z JavaScriptem", RokWydania = 2017,
Gatunek = 3, Cena =(float)29.40 },

           };
            var gatunki = new[] {
                new Gatunek{Id=1,Nazwa="Literatura piekna" },
                new Gatunek{Id=2,Nazwa=" PRzygodowa" },
                new Gatunek{Id=3,Nazwa="Programowanie" },
                new Gatunek{Id=4,Nazwa="Projektowanie stron WWW" },


            };
            var test9 = (from gatunek in gatunki
                         join ksiazka in ksiazki
                         on gatunek.Id equals ksiazka.Gatunek
                         into lj
                         from tmp in lj.DefaultIfEmpty()
                         select new
                         {
                             Gatunek = gatunek.Nazwa,
                             Cena = tmp?.Cena ?? 0,
                         })
                        .GroupBy(x => x.Gatunek)
                        .Select(x => new
                        {
                            Gatunek = x.Key,
                            Liczba = x.Sum(i => i.Cena)
                        });
            Console.WriteLine(  "@@@@@@@@@@@@@@@@");
            foreach (var item in test9)
            {
                Console.WriteLine(item);
            }
            //Stwórz klasy na podstawie przykładów: fota2
            //zadanie 10
            var test10 = (from gatunek in gatunki
                          join ksiazka in ksiazki
                          on gatunek.Id equals ksiazka.Gatunek
                          into lj
                          from tmp in lj.DefaultIfEmpty()
                          select new
                          {
                              gatunek = gatunek.Nazwa,
                              Tytul = tmp?.Tytul ?? "brak",

                          }).GroupBy(x => x.gatunek).Select(x => new
                          {
                              gatunek = x.Key,
                              Ilosc = x.Count(i=> i.Tytul !="brak")
                          });
            Console.WriteLine("@@@@@@2@@@@@@@@@@");
            foreach (var item in test10)
            {
                Console.WriteLine(item);
            }


            //C#: Napisz zapytanie LINQ, które dla dowolnej policzalnej kolekcji punktów
dwuwymiarowych określi ile z tych
            //punktów znajduje się powyżej funkcji sinus. Punkty generuj jako wartości
zmiennoprzecinkowe. Zakres wartości
            //dla współrzędnej X to -10.0 : 10.0. Zakres wartości dla współrzędnej Y określ
samodzielnie. Zademonstruj działanie
            //tego zapytania. Punkty mogą być reprezentowane w dowolny sposób.
            //zadanie2
            var generator = new Random();

                var p1 = new Punkt((float)generator.NextDouble() + (float)generator.Next(-9,
9), (float)generator.NextDouble() + (float)generator.Next(-9, 9));
                Console.WriteLine(p1.x + " , " + p1.y);
                var p2 = new Punkt((float)generator.NextDouble() + (float)generator.Next(-9,
9), (float)generator.NextDouble() + (float)generator.Next(-9, 9));
                Console.WriteLine(p2.x + " , " + p2.y);
                var listADAM = new List<Punkt>();

            listADAM.Add(p2);
            listADAM.Add(p1);
```

```
            listADAM.Add(p2);

                Console.WriteLine(listADAM.FindAll((Punkt) => (Punkt.y >
Math.Sin(Punkt.x))).Count);

            //C#: Napisz funkcję generującą leniwą kolekcję punktów w przestrzeni
trójwymiarowej. Każda składowa powinna być losową
            //liczbą całkowitą z przedziału -20 do 20. Punkty możesz reprezentować w dowolny
sposób np. jako tablicę lub obiekty klasy
            //Tuple. Następnie napisz funkcję, która pozwoli przerwać generowanie tych
wartości, gdy pojawi się punkt, którego trzecia
            //składowa będzie ujemna oraz funkcję zwracającą punkt, którego poszczególne
składowe będą sumą składowych wygenerowanych punktów.
            //Zademonstruj ich działanie.//zz13
            var lista3d = Punkt3d.Losuj();
            Console.WriteLine("xxxxxxxxxxxxxxxxxxxxxxx");
            foreach (var item in lista3d)
            {
                Console.WriteLine(item.X+" "+item.Y+" "+item.Z);
            }


            // C#: Napisz funkcję generującą leniwą kolekcję punktów w przestrzeni
trójwymiarowej. Każda
            // składowa powinna być losową liczbą całkowitą z przedziału -20 do 20. Punkty
możesz reprezentować w dowolny sposób np.
            // jako tablicę lub obiekty klasy Tuple. Następnie napisz funkcję, która pozwoli
przerwać generowanie tych wartości, gdy pojawi
            // się punkt, którego pierwsza składowa będzie dodatnia oraz funkcję, która
określi ile takich punktów było. Zademonstruj ich działanie.



            //C#: Napisz funkcję obliczającą wartość silni w taki sposób, aby możliwe było jej
wywołanie w następujący sposób: 100.Silnia()
            //zadanie 3

            Console.WriteLine(4.silnia());
            //2. C#: Stwórz generyczną klasę Lista implementującą listę łączoną i
przechowującą wartość dowolnego typu. Stwórz ją w taki sposób,
            //aby każdy węzeł był elementem tylko do odczytu. Następnie napisz w tej klasie
metodę, która policzy ile wartości dodatnich jest na tej
            //liście. Napisz również kod, który demonstruje jej działanie.
            //zadanie4
            var mojaLista2 = new MojaLista<float>();
            mojaLista2.Dodaj((float)2.1);
            mojaLista2.Dodaj((float)2.3);
            mojaLista2.Dodaj((float)-2.3);
            mojaLista2.Wyswietl();
            mojaLista2.Policz();

            //C#: Stwórz klasy na podstawie przykładów: fota4 Napisz zapytanie Linq, które dla
każdego autora określi łączną cenę książek, które napisał.
            var autorzy = new[] {
             new Autor { Id=1,Imie="Adam",Nazwisko="Mickiewicz"},
             new Autor { Id=2,Imie="Henryk",Nazwisko="Sienkiewicz"},
             new Autor { Id=3,Imie="Boleslaw",Nazwisko="Prus"},
             new Autor { Id=4,Imie="Enrico",Nazwisko="Buonanno"},
             new Autor { Id=5,Imie="Luis",Nazwisko="Atencio"},
             new Autor { Id=6,Imie="Robert C.",Nazwisko="Martin"},
            };
            var ksiazki2 = new[] {

           new Ksiazka2 {Tytul="Pan Tadeusz",RokWydania=1998,Autor=1,Cena=30 },
            new Ksiazka2 { Tytul = "Potop", RokWydania = 1991, Autor = 2, Cena = 40 },
            new Ksiazka2 { Tytul = "W pustyni i w puszczy", RokWydania = 1990, Autor = 2, Cena
= 30 },
            new Ksiazka2 { Tytul = "Lalka", RokWydania = 1990, Autor = 2, Cena = 50 },
            new Ksiazka2 { Tytul = "Programowanie funkcyjne w C#", RokWydania = 2019, Autor =
5, Cena =(float)71.20 },
            new Ksiazka2 { Tytul = "Programowanie funkcyjne z JavaScriptem", RokWydania =
2017, Autor = 6, Cena =(float)29.40 },

           };
            var test11 = (from autor in autorzy
                          join ksiazka in ksiazki2
                          on autor.Id equals ksiazka.Autor
```

```
                        into lj
                        from tmp in lj.DefaultIfEmpty()
                        select new {
                        Autorr = autor.Imie + autor.Nazwisko,
                        Cena = tmp?.Cena ?? 0

                        }).GroupBy(x=> x.Autorr).Select(x=> new {
                        autorr= x.Key,
                        cena =x.Sum(x=>x.Cena )

                        });
            Console.WriteLine("@@@@@@3@@@@@@@@@@");
            foreach (var item in test11)
            {
                Console.WriteLine(item);
            }

            //C#: Stwórz generyczną klasę Lista implementującą listę łączoną i przechowującą
wartość dowolnego typu. Stwórz ją w taki sposób,
            //aby każdy węzeł był elementem tylko do odczytu. Następnie napisz metodę, która
będzie poszukiwała elementu najmniejszego
            //(dla dowolnego typu). Zademonstruj działanie klasy
            //zadanie5
            var mojaLista23 = new MojaLista<float>();
            mojaLista23.Dodaj((float)2.1);
            mojaLista23.Dodaj((float)2.3);
            mojaLista23.Dodaj((float)-2.3);
            mojaLista23.Wyswietl();
            mojaLista23.ZnajdzNajmniejszy();
            //C#: Stwórz klasę Samochod zawierającą pola Marka, Model, PojemnoscSilnika.
Zaimplementuj w niej interfejs IComparable,
            //który pozwoli sortować instancje tej klasy wg. właściwości PojemnosciSilnika w
kolejności malejącej.
            //zadanie6
            var listaSamochodow = new List<Samochod>();
            listaSamochodow.Add(new Samochod("Audi", "a2", 1200));
            listaSamochodow.Add(new Samochod("Bmw", "m3", 3000));
            listaSamochodow.Add(new Samochod("Bmw", "m4", 3500));
            listaSamochodow.Sort();
            Console.WriteLine("posortowane samochody");
            for (int i = 0; i < listaSamochodow.Count; i++)
            {
                Console.WriteLine(listaSamochodow[i].Marka+" "+ listaSamochodow[i].Modell+" "+
listaSamochodow[i].PojemnoscSilnika);
            }

            //Napisz zapytanie LINQ, które dla dowolnej policzalnej kolekcji punktów
dwuwymiarowych określi ile punktów znajduje się w
            //poszczególnych ćwiartkach układu współrzędnych. Wyniki wyświetl w kolejności
określonej numerem ćwiartki malejąco. Punkty
            //mogą być reprezentowane w dowolny sposób.
            //zadanie7
            Random generator2 = new Random();
            var p3 = new Punkt((float)generator.NextDouble() + (float)generator.Next(-9, 9),
(float)generator.NextDouble() + (float)generator.Next(-9, 9));
            Console.WriteLine(p1.x + " , " + p1.y);
            var p4 = new Punkt((float)generator.NextDouble() + (float)generator.Next(-9, 9),
(float)generator.NextDouble() + (float)generator.Next(-9, 9));
            Console.WriteLine(p2.x + " , " + p2.y);
            var list2 = new List<Punkt>();
            list2.Add(p3);
            list2.Add(p4);
            list2.Add(p3);
            list2.Add(p4);
            var p11 = new Punkt((float)1.1, (float)1.1);
            var p12 = new Punkt((float)-1.1, (float)1.1);
            var p13 = new Punkt((float)-1.1, (float)-1.1);
            var p14 = new Punkt((float)1.1, (float)-1.1);
            list2.Add(p13);
            list2.Add(p14);
            list2.Add(p12);
            list2.Add(p11);
            // Console.WriteLine("ilosc w 1 cwiartce: "+ list2.FindAll(x => (x.x >= 0.0 && x.y
< 0.0)).Count );
            // Console.WriteLine("ilosc w 4 cwiartce: "+ list2.FindAll(x => (x.x >= 0.0 && x.y
>= 0.0)).Count );
```

```
            ///  Console.WriteLine("ilosc w 2 cwiartce: "+ list2.FindAll(x => (x.x < 0.0 &&
x.y >= 0.0)).Count );
            // Console.WriteLine("ilosc w 3 cwiartce: "+ list2.FindAll(x => (x.x < 0.0 && x.y
< 0.0)).Count );



            var test = list2.Select(p1 => new
            {
                Cwiartka = p1.x > 0 && p1.y > 0 ? "pierwsza" : p1.x < 0 && p1.y > 0 ? "druga"
: p1.x < 0 && p1.y < 0 ?
                "trzecia" : p1.x > 0 && p1.y < 0 ? "czwarta" : "0"

            }).GroupBy(i => i.Cwiartka).Select(i => new { i.Key, Liczba = i.Count()
}).OrderByDescending(x=>x.Liczba);

            foreach (var item in test)
            {
                Console.WriteLine(item);
            }

            //C#: Napisz zapytanie LINQ, które dla tablicy dowolnych wyrazów, określi ile było
wyrazów zaczynających się na każdą z liter.
            //zadanie8
            var listaSlow2 = new List<string>();
            listaSlow2.Add("ala");
            listaSlow2.Add("bla");
            listaSlow2.Add("abla");
            listaSlow2.Add("cbla");
            listaSlow2.Add("dcdbla");
            listaSlow2.Add("xdcdbla");
            var test2 = listaSlow2.Select(x => new
            {
                litera = x[0].ToString().ToUpper()
            }).GroupBy(i=>i.litera).Select(x=>new { litera = x.Key , liczba = x.Count() });

            foreach (var item in test2)
            {
                Console.WriteLine(item);
            }

            //2. Napisz funkcję generującą leniwą kolekcję punktów w przestrzeni
trójwymiarowej. Każda składowa powinna być losową liczbą całkowitą
            //z przedziału -20 do 20. Punkty możesz reprezentować w dowolny sposób np. jako
tablicę lub obiekty klasy Tuple. Następnie napisz funkcję,
            //która pozwoli przerwać generowanie tych wartości, gdy pojawi się punkt, którego
trzecia składowa będzie ujemna oraz funkcję zwracającą
            //punkt, którego poszczególne składowe będą sumą składowych wygenerowanych
punktów. Zademonstruj ich działanie.

            //C#: Stwórz klasy na podstawie przykładów: fota5 Napisz zapytanie Linq, które dla
autora określi cenę najtańszej książki, którą napisał.
            //Jeżeli autor nie napisał książek zapytanie powinno zwrócić 0.

        /*   var test12 = (from autor in autorzy
                           join ksiazka in ksiazki2
                           on autor.Id equals ksiazka.Autor
                           into lj
                           from tmp in lj.DefaultIfEmpty()
                           select new {
                               autorr = autor.Imie = autor.Nazwisko,
                               cena = tmp?.Cena ?? 0

                           }).GroupBy(x => x.autorr).Select(x => new {
                               Autorr = x.Key,
                               cena = x.Min(x => x.cena)


                           });*/
            var test13 = (from autor in autorzy
                          join ksiazka in ksiazki2
                          on autor.Id equals ksiazka.Autor
                          into lj
                          from tmp in lj.DefaultIfEmpty()
                          select new {
                          autor = autor.Id,
                          cena = tmp?.Cena ?? 0
```

```
                        }).GroupBy(x=>x.autor).Select(x=>new {
                        autor = x.Key,
                        cena = x.Min(x=>x.cena != 0 )

                        });

            Console.WriteLine("@@@@@@4@@@@@@@@@@");
            foreach (var item in test13)
            {
                Console.WriteLine(item);
            }
            //Napisz zapytanie LINQ, które dla dwóch tablic wyrazów określi ile razy w obu
tablicach ten sam wyraz wystąpił na tej samej pozycji.

           // var test = list1.Zip(list2, (l1, l2) => new { rowne = (l1 == l2) }).Where(x =>
x.rowne == true).Count();

            // select - mapowanie tworzenie nowych elementow
            // groupby - grupowanie zliczanie elementow
            //orderby - sortowanie lementow
            //findall - znjadz wszytkie
            //where - warunek
            //zip mapowanie 2 list
            //Min
            //Average
            //Max
            //foreach


        }
    }

}
```