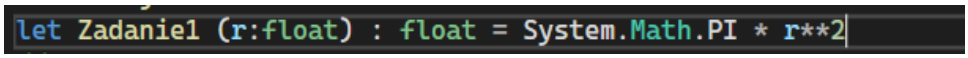
F#:

Lista1:

Zadanie 1.1
Napisz funkcję obliczającą pole koła

let Zadanie1 (r:float) : float = System.Math.PI * r**2



Zadanie 1.2

Napisz funkcję obliczającą wartość pierwiastków równania kwadratowego

Zadanie 1.3
Napisz funkcję, która sprawdzi czy z trzech podanych wartości rzeczywistych, da się zbudować trójkąt.

```
let Zadanie3 (a:float) (b:float) (c:float) =
   if(a + b > c) then
      if(a + c > b) then
         if(c + b > a) then
            printfn "Mozna zbudowac trojkat"
            else
               printfn "Nie mozna zbudowac trojkatu"
      else
         printfn "Nie mozna zbudowac trojkatu"
   else
      printfn "Nie mozna zbudowac trojkatu"
```

```
let Zadanie3 (a:float) (b:float) (c:float) =
    if(a + b > c) then
        if(a + c > b) then
            if(c + b > a) then
                printfn "Mozna zbudowac trojkat"
                else
                    printfn "Nie mozna zbudowac trojkatu"
        else
            printfn "Nie mozna zbudowac trojkatu"
    else
        printfn "Nie mozna zbudowac trojkatu"
```

Zadanie 1.4

Napisz funkcję, która obliczy pole trójkąta, na podstawie długości jego boków. Jeżeli z podanych wartości nie da się zbudować trójkąta rzuć wyjątek z odpowiednią wiadomością

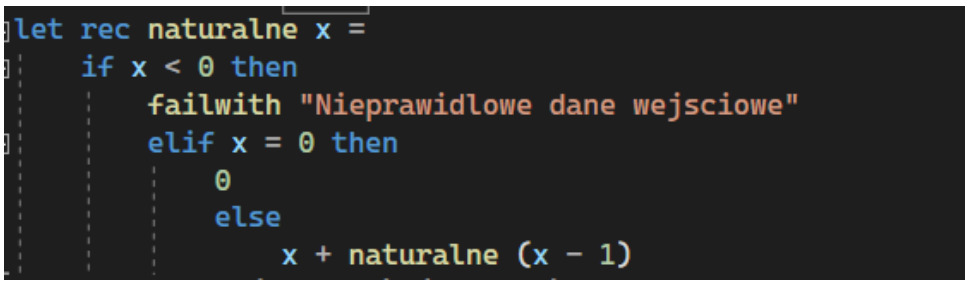
```
let Zadanie4 (a:float) (b:float) (c:float) =

   if(a + b > c) then

      if(a + c > b) then

         if(c + b > a) then

            let p = (a + b + c) / 2.0

            System.Console.WriteLine($"Pole trojkata wynosi: { sqrt(p * (p - a)*(p - b)*(p - c))}")

            else

               printfn "Nie mozna zbudowac trojkatu"

      else

         printfn "Nie mozna zbudowac trojkatu"


   else

      printfn "Nie mozna zbudowac trojkatu"
```

```
let Zadanie4 (a:float) (b:float) (c:float) =
    if(a + b > c) then
        if(a + c > b) then
            if(c + b > a) then
                let p = (a + b + c) / 2.0
                System.Console.WriteLine($"Pole trojkata wynosi: { sqrt(p * (p - a)*(p - b)*(p - c))}")
                else
                    printfn "Nie mozna zbudowac trojkatu"
        else
            printfn "Nie mozna zbudowac trojkatu"

    else
        printfn "Nie mozna zbudowac trojkatu"
```

Zadanie 1.5

Napisz funkcję, która rekurencyjnie oblicza sumę n pierwszych liczba naturalnych

```
let rec naturalne x =
  if x < 0 then
    failwith "Nieprawidlowe dane wejsciowe"
    elif x = 0 then
      0
      else
        x + naturalne (x - 1)
```



Zadanie 1.6

Napisz funkcje, która rekurencyjnie wyznaczy wartość x^n, gdzie x i n są dowolnymi liczbami naturalnymi

```
let rec zadanie16r x n =
  if x <= 0 then
    failwith "Niepoprawne x"
  else
    if n < 0 then
      failwith "Niepoprawne n"
    elif n = 0 then
      1
    else
      x*zadanie16r x (n - 1)
```

```
let rec zadanie16r x n =
    if x <= 0 then
        failwith "Niepoprawne x"
    else
        if n < 0 then
            failwith "Niepoprawne n"
        elif n = 0 then
            1
        else
            x*zadanie16r x (n - 1)
```

Zadanie 1.7

Napisz funkcję, która rekurencyjnie oblicza sumę n pierwszych liczb naturalnych

```
let rec zadanie17 x =
  match x with
  | 0 | 1 -> x
  | x -> zadanie17 (x - 1) + zadanie17 (x - 2)
```

```
let rec zadanie17 x =
    match x with
    | 0 | 1 -> x
    | x -> zadanie17 (x - 1) + zadanie17 (x - 2)
```

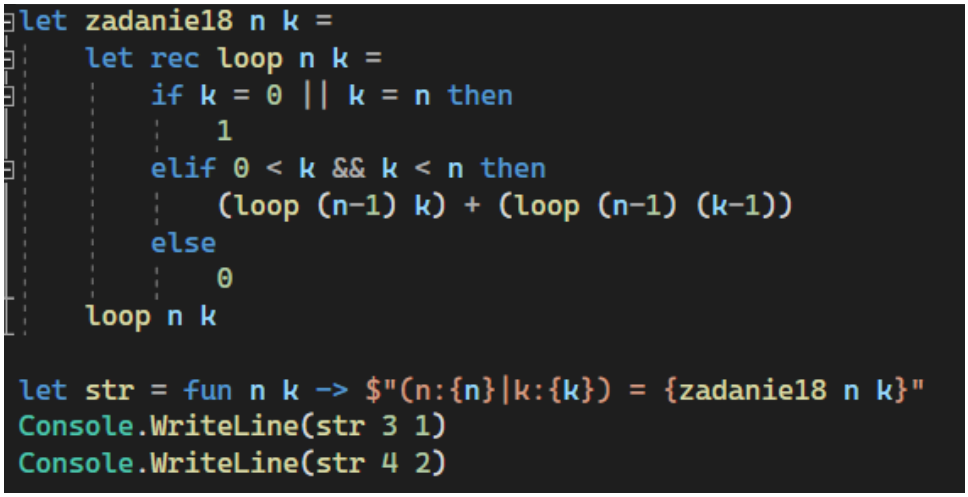
Zadanie 1.8

Napisz funkcję, która dla dowolnych liczb całkowitych n i k obliczy wartość dwumiany Newtona: $\left(_{k=\frac{n!}{k!(n-k)!}}\right)$ **zdefiniowanego w sposób rekurencyjny**

$$\binom{n}{k} = \begin{cases} 1 & \text{dla } k = 0 \\ 1 & \text{dla } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{dla } k \neq 0 in \neq k \end{cases}$$

```
let zadanie18 n k =
  let rec loop n k =
    if k = 0 || k = n then
      1
    elif 0 < k && k < n then
      (loop (n-1) k) + (loop (n-1) (k-1))
```

```
    else
        0
loop n k

let str = fun n k -> $"(n:{n}|k:{k}) = {zadanie18 n k}"
Console.WriteLine(str 3 1)
Console.WriteLine(str 4 2)
```



Zadanie 1.9

Napisz funkcję, która rekurencyjnie sprawdza czy dana liczba jest liczbą pierwszą.

```
let rec zadanie19 x =
    let i = 2
    if x = i then
        printfn "Koniec funkcji"
    else
        let j = 2
        if x%i = 0 then
            printfn "Liczba %d nie jest liczba pierwsza" (i + 1)
            else
                printfn "Liczba %d jest liczba pierwsza" (i + 1)
```

```
let rec zadanie19 x =
    let i = 2
    if x = i then
        printfn "Koniec funkcji"
    else
        let j = 2
        if x%i = 0 then
            printfn "Liczba %d nie jest liczba pierwsza" (i + 1)
            else
                printfn "Liczba %d jest liczba pierwsza" (i + 1)
```

Zadanie 1.10

Napisz funkcję, która na podstawie 1000 rzutów kostką do gry określi prawdopodobieństwo wyrzucenia szóstki.

```
let rec zadanie110 i xd =
  let szesc = rand.Next(1, 7)
  if i = 0 then
    xd/1000.0
  else
    if szesc = 6 then
      zadanie110 (i - 1) (xd + 1.0)
    else
      zadanie110 (i - 1) xd
```

```
let rec zadanie110 i xd =
    let szesc = rand.Next(1, 7)
    if i = 0 then
        xd/1000.0
    else
        if szesc = 6 then
            zadanie110 (i - 1) (xd + 1.0)
        else
            zadanie110 (i - 1) xd
```

Zadanie 1.11

Napisz funkcję, która na podstawie 1000 rzutów dwiema kostkami do gry określi prawdopodobieństwo wyrzucenia dwóch szóstek.

```
et rec zadanie111 i xd =
  let szesc = rand.Next(1, 7)
  let szesc2 = rand.Next(1, 7)
  if i = 0 then
```

```
    xd/1000.0
  else
    if szesc = 6 && szesc2 = 6 then
      zadanie111 (i - 1) (xd + 1.0)
    else
      zadanie111 (i - 1) xd
```

```
let rec zadanie111 i xd =
    let szesc = rand.Next(1, 7)
    let szesc2 = rand.Next(1, 7)
    if i = 0 then
        xd/1000.0
    else
        if szesc = 6 && szesc2 = 6 then
            zadanie111 (i - 1) (xd + 1.0)
        else
            zadanie111 (i - 1) xd
```

Zadanie 1.12

Napisz funkcję, określającą największy wspólny dzielnik

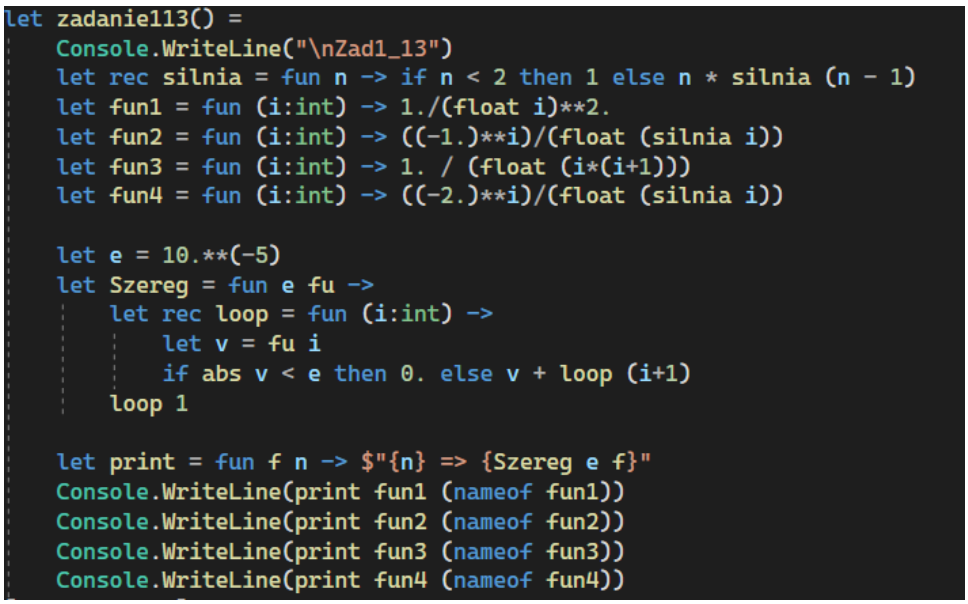
Zadanie 1.13

Oblicz przybliżoną wartość szeregu nieskończonego. Zatrzymaj obliczenia kiedy wartość bezwzględna kolejnego elementu w szeregu będzie mniejsza niż ustalona dokładność e (np. $e = 10^{-7}$):

1. $\sum_{i=1}^{\infty} \frac{1}{i^2}$
2. $\sum_{i=1}^{\infty} \frac{(-1)^i}{i!}$
3. $\sum_{i=1}^{\infty} \frac{1}{i(i+1)}$
4. $\sum_{i=1}^{\infty} \frac{(-2)^i}{i!}$

```
let zadanie113() =
  Console.WriteLine("\nZad1_13")
  let rec silnia = fun n -> if n < 2 then 1 else n * silnia (n - 1)
  let fun1 = fun (i:int) -> 1./(float i)**2.
```

```
let fun2 = fun (i:int) -> ((-1.)**i)/(float (silnia i))
let fun3 = fun (i:int) -> 1. / (float (i*(i+1)))
let fun4 = fun (i:int) -> ((-2.)**i)/(float (silnia i))

let e = 10.**(-5)
let Szereg = fun e fu ->
   let rec loop = fun (i:int) ->
      let v = fu i
      if abs v < e then 0. else v + loop (i+1)
   loop 1

let print = fun f n -> $"{n} => {Szereg e f}"
Console.WriteLine(print fun1 (nameof fun1))
Console.WriteLine(print fun2 (nameof fun2))
Console.WriteLine(print fun3 (nameof fun3))
Console.WriteLine(print fun4 (nameof fun4))
```



```
let zadanie113() =
    Console.WriteLine("\nZad1_13")
    let rec silnia = fun n -> if n < 2 then 1 else n * silnia (n - 1)
    let fun1 = fun (i:int) -> 1./(float i)**2.
    let fun2 = fun (i:int) -> ((-1.)**i)/(float (silnia i))
    let fun3 = fun (i:int) -> 1. / (float (i*(i+1)))
    let fun4 = fun (i:int) -> ((-2.)**i)/(float (silnia i))

    let e = 10.**(-5)
    let Szereg = fun e fu ->
        let rec loop = fun (i:int) ->
            let v = fu i
            if abs v < e then 0. else v + loop (i+1)
        loop 1

    let print = fun f n -> $"{n} => {Szereg e f}"
    Console.WriteLine(print fun1 (nameof fun1))
    Console.WriteLine(print fun2 (nameof fun2))
    Console.WriteLine(print fun3 (nameof fun3))
    Console.WriteLine(print fun4 (nameof fun4))
```

Zadanie 1.14

Napisz powyższe funkcje z wykorzystaniem rekurencji ogonowej

XD

Zadanie 1.15

Napisz funkcję zamieniającą temperaturę wyrażoną w stopniach Celcjusza na temperaturę wyrażoną w stopniach Fahrenheita zgodnie z poniższym

$$T_{Fahrenheit} = 32 + \frac{9}{5} \cdot T_{Celsius}$$

wzorem:

```
let zadanie115 c =
   32.0<F> + (9.0<F>/5.0<C>) * c
```

Zadanie 1.16

Odwrotność zadania1.15

```
let zadanie116 f =
   5.0<C>/9.0<F> * (f-32.0<F>)
```

Zadanie 1.17

Napisz funkcję, która sprawdzi czy dane słowo jest palindromem

```
printfn"Zadanie 1.17"
   let ZnajdzPalindrom (wyraz:string)=
      for i=0 to wyraz.Length-1 do
       if(wyraz.[i] = wyraz.[wyraz.Length-i-1])then
         System.Console.Write(wyraz.[i])
        else
         System.Console.Write("to nie paliandrom")
   //ZnajdzPalindrom "anna"
   ZnajdzPalindrom "bartek"
```

Zadanie 1.18

Napisz funkcję, która policzy ile razy w podanym tekście wystąpił określony znak

```
//Zadanie 1.18
```

```
printfn""
printfn"Zadanie 1.18"
let znajdzZnak (slowo:string)(znak:char)=
    let rec petla i licznik=
        if i<slowo.Length then
            if slowo.[i] = znak then
                petla (i+1) (licznik+1)
            else petla (i+1) licznik
        else
            licznik
    petla 0 0

System.Console.WriteLine(znajdzZnak "anana" 'a')
```

Zadanie 1.19

Napisz funkcję, która określi liczbę wyrazów w podanym tekście

```
//Zadanie1.19
    printfn"Zadanie 1.19"
    let policzSlowa (tekst:string)=
        let rec petla i licznik=
            let spacja:char=' '
            if i<tekst.Length then
                if tekst.[i] = spacja then
                    petla (i+1) (licznik+1)
                else petla (i+1) licznik
            else
                licznik+1
        petla 0 0
    System.Console.WriteLine(policzSlowa "anna ma kota")
```

Zadanie 1.20

Napisz funkcję, która określi liczbę cyfr występujących pod rząd w tekście.

```
printfn"Zadanie 1.20"
   let policzCyfry (tekst:string)=
      let rec petla i licznik=
         if i<tekst.Length then
            if(
              (tekst.[i]='0' || tekst.[i]='1' || tekst.[i]='2' || tekst.[i]='3'
               ||tekst.[i]='4' || tekst.[i]='5' || tekst.[i]='6'||tekst.[i]='7'||
               tekst.[i]='8'||tekst.[i]='9')
               &&
               (tekst.[i+1]='0' || tekst.[i+1]='1'
               || tekst.[i+1]='2' || tekst.[i+1]='3'
               ||tekst.[i+1]='4' || tekst.[i+1]='5' || tekst.[i+1]='6'||tekst.[i+1]='7'||
                  tekst.[i+1]='8'||tekst.[i+1]='9')
             )
            then petla (i+1) (licznik+1)
            else petla (i+1) licznik
         else
            licznik
      petla 0 0
   System.Console.WriteLine(policzCyfry "qwer23aafdfdasfsd34tyuy45uy")
```

Zadanie 1.21

Napisz aplikację konsolową, która pozwoli użytkownikowi na wprowadzenie imienia i nazwiska

```
let zadanie121() =
   printfn "Prosze podac imie i nazwisko"
   let znaki = Console.ReadLine()
   Console.WriteLine($"Witaj {znaki}")
```

```fsharp
let zadanie121() =
    printfn "Prosze podac imie i nazwisko"
    let znaki = Console.ReadLine()
    Console.WriteLine($"Witaj {znaki}")
```

Zadanie 1.22

Napisz aplikację konsolową, który pozwoli użytkownikowi na wprowadzenie roku. W odpowiedzi program powinien wyświetlić informacje czy rok jest to rok przystępny czy nie

```
let zadanie122() =
   printfn "Prosze podac rok"
   let rok  = Console.ReadLine()
   if int rok%4 = 0 then
      printfn "Rok jest przestepny"
   else
      printfn "Rok nie jest przestepny"
```

```fsharp
let zadanie122() =
    printfn "Prosze podac rok"
    let rok  = Console.ReadLine()
    if int rok%4 = 0 then
        printfn "Rok jest przestepny"
    else
        printfn "Rok nie jest przestepny"
```

Zadanie 1.23

Napisz aplikację konsolowa, która na podstawie podanych użytkownika 3 liczb rzeczywistych wyświetli na ekranie komunikat, czy da się z nich utworzyć trójkąt równoboczny, równoramienny, prostokątny, dowolny inny lub że z podanych wartości nie da się utworzyć trójkąta

Zadanie 1.24

Napisz aplikację konsolowa, która przyjmie od użytownika ciąg 11 cyfr. Sprawdź, czy cyfry te tworzą poprawny numer PESEL, oraz na jego podstawie wyświetl informacje o dacie urodzenia danej osoby oraz czy jest to kobieta czy mężczyzna.

Zadanie 1.25

Napisz aplikację konsolową, która przyjmie od użytkownika dowolny tekst, i wyświetli go zakodowanego zgodnie z szyfrem cezara

Zadanie 1.26

Napisz aplikację konsolowa, która przyjmie od użytkownika dowolny tekst zakodowany zodnie z szyfrem Cezara, i odkoduje go.

Zadanie 1.27

Napisz aplikację konsolowa, która wczytuje liczbę całkowitą określającą liczbę minut od północy. Program powinien wyświetlać w odpowiedzi konkretną godzinę, która będzie odpowiadać tej piczbie. Jeżeli podana liczba godzin będzie przekraczała liczbę godzin w ciągu doby na ekranie powinien pojawić się stosowny komunikat

```
let zadanie127() =
    let Czas = fun () ->
        Console.WriteLine($"Podaj liczbe minut:")
        let i = try int (Console.ReadLine() |> uint) with | _ -> -1
        if i < 0 || i > 24*60 then
            Console.WriteLine($"Poza zakresem (max {24*60})")
        else
            Console.WriteLine($"Czas {int (i/60)}:{i%60}")
    Czas()
```

```
let zadanie127() =
    let Czas = fun () ->
        Console.WriteLine($"Podaj liczbe minut:")
        let i = try int (Console.ReadLine() |> uint) with | _ -> -1
        if i < 0 || i > 24*60 then
            Console.WriteLine($"Poza zakresem (max {24*60})")
        else
            Console.WriteLine($"Czas {int (i/60)}:{i%60}")
    Czas()
```

Zadanie 1.28

Napisz aplikację konsolowa, która wczytuje liczbę całkowitą określającą liczbę minut do startu. Program w odpowiedzi powinien wyświetlać komunikaty: “Do startu pozostało … minut”. Jeżeli odliczanie dojdzie do zera powinien wyświetlić się odpowiedni komunikat. Odpowiedzią funkcję proszę umieścić w osobnych pliku.

Zadanie 1.29

Napisz aplikację konsolowa, która rozwiązuje problem Collatza. Program

powinien wczytywać od użytkownika liczbę całkowitą dodatnią. Na jej podstawie obliczamy kolejne wartości:

$$C_{n+1} = \begin{cases} 0.5 * C_n & \text{jeżeli } C_n \text{ jest parzysta} \\ 3 * C_n + 1 & \text{w przeciwnym przypadku} \end{cases}$$

Zadanie 1.30

Napisz aplikację konsolową, która pozwoli na wczytywanie z klawiatury liczba całkowitych. Oblicz średnią tych liczb. Użytkownik powinien wprowadzać wartości dopóki nie poda wartości ujemnej. Odpowiednią funkcję proszę umieścić w osobnym pliku.

```
let zadanie130() =
   let Prog = fun ()->
      let rec loop = fun sum num ->
         Console.WriteLine($"Podaj liczbe:")
         let i = try int (Console.ReadLine() |> uint) with | _ -> -1
         if i > 0 then
            Console.WriteLine($"Srednia: {(sum+i)}/{(num+1)} => {(sum+i)/(num+1)}")
            loop (sum+i) (num+1)
         else
            ()
      loop 0 0
   Prog()
```

Lista2:

Zadanie 2.1

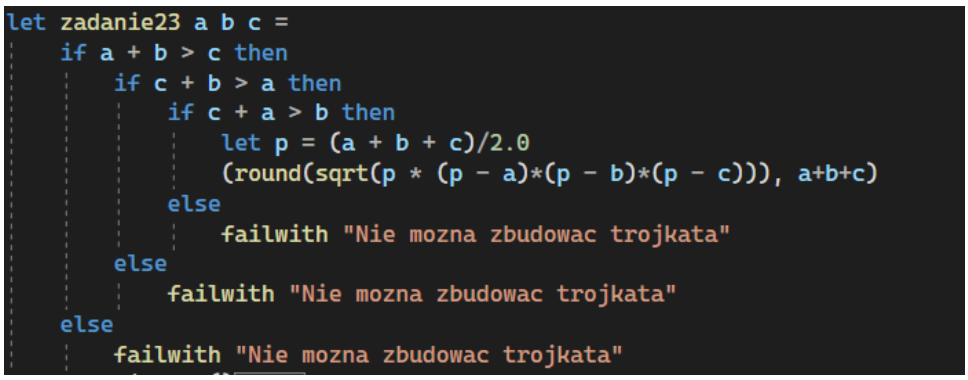
Zadanie 2.2

Napisz funkcję, która będzie wczytywała od użytkownika dwie liczby, zwracała je w formie pary. Następnie wykorzystaj dopasowanie wzorców do krotki i wyświetl na ekranie informacje czy pierwsza liczba z pary jest większa niż druga, czy druga jest większa czy obie są równe

Zadanie 2.3

Napisz funkcję, która jako parametry będzie przyjmowała długości trzech boków trójkąta oraz zwracała jego pole i obwód

```
let zadanie23 a b c =
    if a + b > c then
        if c + b > a then
            if c + a > b then
                let p = (a + b + c)/2.0
                (round(sqrt(p * (p - a)*(p - b)*(p - c))), a+b+c)
            else
                failwith "Nie mozna zbudowac trojkata"
        else
            failwith "Nie mozna zbudowac trojkata"
    else
        failwith "Nie mozna zbudowac trojkata"
```



Zadanie 2.4

Napisz funkcje, która będzie przyjmowała jako parametr łańcuch znaków określających adres email i wydzielała z niego identyfikator użytkownika oraz adres domenowy serwera

```
let zadanie24() =
    let email_slice (email)  =
        let len = String.length email
        let rec find i c = if len <= i || email[i] = c then i else find (i+1) c
        let i = find 0 '@'
        (email.[0..i-1], email.[i+1..len])

    let fu = fun a -> $"{a} => {email_slice a}"
    Console.WriteLine(fu "user@example.com")
```

Zadanie 2.5

Wykorzystaj funkcję napisaną w poprzednim zadaniu do podziału adresu na części, a następnie wyświetl informacje czy podany adres należy do domeny PCz czy nie.

```
let zadanie25() =
    let pcz_email email =
        match Zad2_4.email_slice email with
        | (_, "pcz.pl") -> "Adres pcz"
        | _ -> "Nie adres pcz"

    let fu = fun a -> $"{a} => {pcz_email a}"
    Console.WriteLine(fu "user@example.com")
    Console.WriteLine(fu "user@pcz.pl")
    Console.WriteLine(fu "user2@example.com")
```

Zadanie 2.6

Zadanie 2.7

Zadanie 2.8

Napisz aplikację, w której zdefiniujesz nowy typ danych opisujących ułamki zwykłe. Napisz funkcję umożliwiające wykonanie podstawowych operacji arytmetycznych: dodawanie, odejmowanie, mnożenie i dzielenie.
Wykorzystaj krotki

Zadanie 2.9

Napisz tę samą aplikację co w zadaniu 2.8 ale z wykorzystaniem rekordów.

```
let zadanie29() =
   type Ulamek =
      {
         L:int
         M:int
      }
      static member (+) (a:Ulamek, b:Ulamek) =
         if b.M = a.M then {L=(a.L+b.L);M=a.M} else {L=(a.L*b.M+b.L*a.M);M=a.M*b.M}
      static member (-) (a:Ulamek, b:Ulamek) =
         if b.M = a.M then {L=(a.L-b.L);M=a.M} else {L=(a.L*b.M+b.L*a.M);M=a.M*b.M}
      static member (~-) (a:Ulamek) =
         {L=(-a.L);M=a.M}
      static member (*) (a:Ulamek, b:Ulamek) =
         {L=a.L*b.L;M=a.M*b.M}
      static member (/) (a:Ulamek, b:Ulamek) =
         {L=a.L*b.M;M=a.M*b.L}
      override this.ToString() = $"[{this.L}/{this.M}]"

   let u1 = {L=1;M=2}
   let u2 = {L=1;M=3}
   let u3 = {L=3;M=1}
   let u4 = {L=2;M=20}
   Console.WriteLine($"{u1}/{u2} = {u1/u2}")
   Console.WriteLine($"{u3}/{u2} = {u3/u2}")
   Console.WriteLine($"{u3}-({u2}/{u1})+{u4} = {u3-(u2/u1)+u4}")
```

Zadanie 2.10

Napisz program, w którym zadeklarujesz nowy typ opisujący datę, a

następnie wykorzystasz go do określenia jaki jest dzień tygodnia w którym ta data wypada.

```
let zadanie210() =
   type Kalendarz =
      {
         D:int
         M:int
         R:int
      }
      member this.Przestepny = if (this.R % 4) = 0 then 1 else 0
      // member this.Dni_wM =
      //    match this with
      //    | {M = 2} -> 28 + this.Przestepny
      //    | {M = m} when (m % 2) = 1 -> 31
      //    | _ -> 30
      // member this.Dni_wR = 365 + this.Przestepny
      member this.Dzien_Roku = (this.M - 1) * 31 - int ((this.M - 1) / 2) + this.D - (if this.M > 2 then 2 - this.Przestepny else 0)
      member this.Dzien_odR0 = (365 * this.R + int (this.R / 4)) + this.Dzien_Roku + this.D
      member this.Dzien_Tyg = ((this.Dzien_odR0 - {D=1;M=1;R=1990}.Dzien_odR0) % 7)
      override this.ToString() = $"{this.D:D2}-{this.M:D2}-{this.R:D4}r"

   let d1 = {D=1;M=12;R=2022}
   let d0 = {D=1;M=1;R=1990}
   let dt = ["pn";"wt";"śr";"cz";"pt";"so";"nd"]
   Console.WriteLine($"{d1} => {dt.[d1.Dzien_Tyg]} \ttest: {DateTime(2022, 12, 1).DayOfWeek}")
   Console.WriteLine($"{d0} => {dt.[d0.Dzien_Tyg]} \ttest: {DateTime(1990, 1, 1).DayOfWeek}")
```

Zadanie 2.11

Napisz funkcję realizującą bezpiecznie dzielenie. Funkcja powinna przyjmować dwie liczby, które chcemy podzielić. Jeżeli jest to możliwe to funkcja powinna zwracać wynik dzielenia, a jeżeli nie komunikat "Nie można dzielić przez 0"

```
let zadanie211() =
    type OpResult =
        | Float of float
        //| Single
        | OpError of string
        static member (/) (a:OpResult, b:OpResult) :OpResult =
            match (a,b) with
            | (Float a,Float b) -> if b = 0 then OpError $"->({a}/{b})<- Error: Dzielimy przez zero!" else Float (a / b)
            | _ -> OpError $"{a}/{b}"
        override this.ToString() =
            match this with
            | Float a -> $"{a}"
            | OpError a -> $"{a}"

    let a = Float 0
    let b = Float 1
    let c = Float 5
    Console.WriteLine($"{b}/{a} = {b/a}")
    Console.WriteLine($"{b}/{c} = {b/c}")
    Console.WriteLine($"{b}/{c}/{c} = {b/c/c}")
    Console.WriteLine($"{b}/{a}/{c} = {b/a/c}")
    Console.WriteLine($"({b}/{a})/({c}/{a}) = {(b/a)/(c/a)}")
```

Zadanie 2.12

Zadanie 2.13

Zadanie 2.14

Zadanie 2.15

Napizs program, który będzie pozwalał przechowywać następujące informacje o osobie: imie, nazwisko, wiek. Przygotuj odpowiedni typ danych. Napisz program, który będzie pozwalał na wprowadzenie informacji o osobie, modyfikwanie ich oraz wyświetlanie. Po uruchomieniu programu użytkownik powinien zobaczyć menu:

```
WYBIERZ OPCJE
1 - utworzenie rekordu
2 - modyfikacja rekordu
3 - pokaz rekord
4 - zakończenie programu
```

Po wybraniu odpowiedniej opcji użytkownik powinien mieć możliwość wprowadzenia danych (program powinien wyświetlać informacje o jakie dane prosi użytkownika), lub ich zobaczenia. Podczas modyfikowania danych jeżeli użytkownik wprowadzi łańcuch pusty (w przypadku imienia lub nazwiska) lub 0 (w przypadku wieku) program powininen zachowywać wcześniejsze dane. Po zakończeniu wprowadzania lub przeglądania danych oraz w przypadku, gdy użytkownik wybierze inną opcję niż od 1 do 4 program powinien wracać do menu głównego. Wykorzystaj dopasowanie wzorców.

Zadanie 2.16 Zmodyfikuj poprzednie zadanie, tak aby wyświetlając dane wprowadzonej osoby pojawiała się również informacja czy jest ona pełnoletnia, czy nie. Wykorzystaj dopasowanie wzorca do rekordu.

(CHuj nie piszę tego pierdol się)

```
let zadanie215 =
    type Osoba =
        {
            Imie:string
            Nawisko:string
            Wiek:uint
        }
        override this.ToString() = $"({this.Imie} {this.Nawisko} {this.Wiek})"
    type Dane =
        | DaneOsoby of Osoba
        | Empty


    let Program () =
        let GetValues = fun (text:string) ->
```

```
        Console.WriteLine(text)
        Console.ReadLine().Split(' ', StringSplitOptions.TrimEntries)
    let GetCmd = fun () ->
        let args = GetValues "Opcje:\n1 - utworzenie rekordu\n2 - edycja rekordu\n3 - pokaz rekord\n4 - zakończ"
        try (args[0] |> int) with | _ -> 0
    let GetOsosba = fun () ->
        let args = GetValues ("Podaj dane rekordu:\n imie nazwisko wiek")
        try DaneOsoby {Imie=args[0];Nawisko=args[1];Wiek=args[2]|>uint}with | _ -> Empty
    let rec CmdLoop = fun (data:Dane) ->
        match GetCmd() with
        | 1 -> CmdLoop(GetOsosba())
        | 2 ->
            match data with
            | DaneOsoby s -> CmdLoop(GetOsosba())
            | Empty ->
                Console.WriteLine("Brak danych, edycja przerwana")
                CmdLoop(data)
        | 3 ->
            Console.WriteLine(data)
            CmdLoop(data)
        | 4 -> Console.WriteLine("Exit")
        | _ ->
            Console.WriteLine("Błędny parametr")
            CmdLoop(data)
    CmdLoop(Empty)
Program ()
```

Zadanie 2.16

Lista3:

Zadanie 3.1

Napisz funkcje, która generuje listę zbudowaną z n pierwszych liczba naturalnych. Sygnatura funkcji powinna przyjmować następującą postać:

```
nPierwszych: n:int->Lista<int>
```

```
let zadanie31() =
    type Lista<'a> =
        | Pusta
        | Wezel of 'a*Lista<'a>
        // odkomentować dla czytelności
        // override this.ToString() =
        //     match this with
        //     | Wezel(v,n) -> $"{v} -> {n}"
        //     | _ -> ""\
        let nPierwszych (n) =
        let rec recur = fun (i, n) -> if i < n then Wezel(i, recur (i+1, n)) else Pusta
        recur (0 ,n)


    Console.WriteLine($"(3)->{nPierwszych 3}")
```

Zadanie 3.3

Napisz funkcję, która zwróci n-ty element listy.

```
let zadanie33() =
    open Zad3_1 // dodaje moduł z listą
    Console.WriteLine("\nZad3_3")
    let rec nElement (l : Lista<'a>, n :int) : ('a) =
        match l with
        | Pusta ->  failwith "n poza zakresem"
        | Wezel(lval, lnext) when n > 1 -> nElement (lnext, n-1)
        | Wezel(lval, lnext) -> lval
```

```
    let list = nPierwszych 5
    Console.WriteLine($"{(list)} \t [3] -> {(nElement (list, 3))}")
```

Zadanie 3.4

Napisz funkcję, która określi czy dany element znajduje się na liście.

```
let zadanie34() =
    open Zad3_1 // dodaje moduł z listą
    Console.WriteLine("\nZad3_4")
    let rec elementIsnieje (l : Lista<'a>, e :'a) : Boolean =
        match l with
        | Pusta ->  false
        | Wezel(lval, lnext) when lval = e -> true
        | Wezel(lval, lnext) -> elementIsnieje(lnext, e)


    let list = nPierwszych 5
    Console.WriteLine($"{(list)} \t [2] -> {(elementIsnieje(list, 2))}, \t [8] ->
{(elementIsnieje(list, 8))}")
```

Zadanie 3.6

Napisz funkcję, która usuwa z listy element na podanej pozycji.

```
let zadanie36() =
    open Zad3_1 // dodaje moduł z listą
    Console.WriteLine("\nZad3_6")
    let rec nUsun (l : Lista<'a>, n :int) : (Lista<'a>) =
        match l with
        | Pusta ->  failwith "n poza zakresem"
        | Wezel(lval, lnext) when n > 1 -> Wezel(lval, nUsun (lnext, n-1)) // przepisujemy
początkowe elementy listy
        | Wezel(lval, lnext) -> lnext // pomijamy element i przekazujemy resztę listy


    let list = nPierwszych 5
```

```
    Console.WriteLine($"{(list)} \t [3] -> {(nUsun (list, 3))}")
```

Zadanie 3.12

Napisz funkcję, która będzie odwracała kolejność elementów na liście.

```
let zadanie312() =
    open Zad3_1 // dodaje moduł z listą
    Console.WriteLine("\nZad3_12")
    let rec nRewers (l : Lista<'a>, lr : Lista<'a>) : (Lista<'a>) =
        match l with
        | Wezel(lval, lnext) ->  nRewers(lnext, Wezel(lval, lr))
        | Pusta ->  lr


    let list = nPierwszych 5
    Console.WriteLine($"{(list)} \t [3] -> {(nRewers (list, Pusta))}")
```

Zadanie 3.14

Napisz funkcję, która będzie przyjmowała dwie listy liczb całkowitych i zwracałą listę wartości logicznych, gdzie true określa, że liczba na pierwszej liście była większa, a false, że wartość na drugiej liście była większa. Jeżeli jedna lista jest dłuższa od drugiej zwróć wyjątek informujący o tym fakcie

```
let zadanie314() =
    open Zad3_1 // dodaje moduł z listą
    open Zad3_12 // dodaje moduł z listą
    Console.WriteLine("\nZad3_14")
    let rec nCompare (la : Lista<int>, lb : Lista<int>) : (Lista<Boolean>) =
        match la, lb with
        | Wezel(laval, lanext), Wezel(lbval, lbnext) ->  Wezel(laval > lbval, nCompare(lanext,
lbnext))
        | Pusta, Pusta  ->  Pusta
        | _  ->  failwith "nierówne listy"


    let list = nPierwszych 5
```

```
    let list2 = nRewers(list, Pusta)
    Console.WriteLine($"({list}) > ({list2}) \t -> {(nCompare (list, list2))}")
```

Zadanie 3.18
Zaimplementuj stos wykorzystując przedstawioną listę łączoną

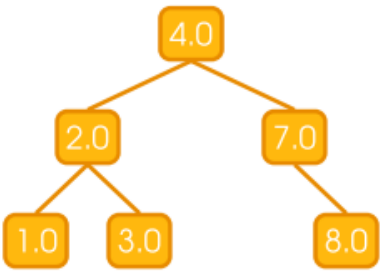
```
let zadanie318() =
    open Zad3_1 // dodaje moduł z listą
    Console.WriteLine("\nZad3_18")


    type Stos<'b> =
      {
        stos: Lista<'b>
      }
      member this.Wartsc() :'b =
        match this.stos with
        | Pusta -> failwith "Pusty stos"
        | Wezel(v,next) -> v


      member this.Dodaj(v:'b) : Stos<'b> =
        {stos = Wezel(v, this.stos)}


      member this.Zdejmij() : Stos<'b> =
        match this.stos with
        | Pusta -> failwith "Pusty stos"
        | Wezel(v,next) -> {stos=next}



    let st = {stos=Pusta}.Dodaj(1).Dodaj(2).Dodaj(3)
    Console.WriteLine($"{st} -> wartos -> {st.Wartsc()}")
    let st2 = st.Zdejmij()
```

```
Console.WriteLine($"{st2} -> wartos -> {st2.Wartsc()}")
```

Zadanie 3.24

Ścieżką do węzła n w drzewie nazywamy kolekcje wszystkich węzłów prowadzących od korzenia do tego węzła. Przykładowo,

Zadanie 3.24 Ścieżką do węzła n w drzewie nazywamy kolekcję wszystkich węzłów prowadzących od korzenia do tego węzła. Przykładowo, dla drzewa z rys. 3.9, ścieżka do węzła 8.0 to $\{4.0, 7.0, 8.0\}$. Napisz funkcję zwracającą ścieżkę do węzła z określoną wartością (dla uproszczenia zakładamy, że wartości w drzewie się nie powtarzają).



Rysunek 3.9: Przykładowe uporządkowane drzewo binarne

```
let zadanie324() =
    Console.WriteLine("\nZad3_24")


    type Drzewo =
        | Puste
        | Wezel of float*Drzewo*Drzewo
        override this.ToString() =
            match this with
            | Wezel(v,l,p) -> $"{v} l: ({l}) p: ({p})"
            | _ -> ""


    let rec Dodaj (v:float, d: Drzewo) :Drzewo =
        match d with
        | Puste -> Wezel(v, Puste, Puste)
        | Wezel(wv, l, p) when wv <= v -> Wezel(wv, l ,Dodaj (v, p) )
        | Wezel(wv, l, p) -> Wezel(wv, Dodaj (v, l) , p )
```

```
let rec Sciezka (v:float, d: Drzewo) : Zad3_1.Lista<float> =
    match d with
    | Puste -> Zad3_1.Pusta
    | Wezel(wv, l, p) when wv <= v -> Zad3_1.Wezel(wv, Sciezka(v, p) )
    | Wezel(wv, l, p) -> Zad3_1.Wezel(wv, Sciezka(v, l) )

let d = Dodaj(5, Dodaj(6,Dodaj(8,Dodaj(-10,Dodaj(2,Dodaj(3, Puste))))))
Console.WriteLine($"{d}")
Console.WriteLine($" scizka {6} -> {Sciezka (6, d)}")
```

Lista4:

Zadanie 4.1

Na poprzednich zajęciach tworzyliśmy listę od podstaw. Napisz funkcję wyższych rzędów mapuj, która będzie dokonywała dowolnego mapowania. Sposób zmiany elementów zdefiniuj jako parametr funkcji mapuj

```
let zadanie41() =
    Console.WriteLine("\nZad4_1")

type Lista<'a> =
    | Pusta
    | Wezel of 'a*Lista<'a>

let nPierwszych (n) =
    let rec recur = fun (i, n) -> if i < n then Wezel(i, recur (i+1, n)) else Pusta
    recur (0 ,n)

/// ------- rozwiązanie
let rec mapuj map_fun = function
```

```
  | Pusta -> Pusta
  | Wezel(v, n) -> Wezel(map_fun v, mapuj (map_fun) n)
  /// ------- rozwiązanie
  let list = nPierwszych (3)
  Console.WriteLine($"{list}")

  let list2 = mapuj (fun x -> x * x) list
  Console.WriteLine($"{list2}")

  let fff = fun x -> x |> float
  let list3 = mapuj fff list
  Console.WriteLine($"{list3}")

  type TTT = {abc:int}
  let list4 = mapuj (fun x -> {abc=x} ) list
  Console.WriteLine($"{list4}")
```

Zadanie 4.2

Napisz funkcję wyższych rzędów, która pozwoli dokonać dowolnej agregacji elementów zapisanych w drzewie.

```
let zadanie42() =
  Console.WriteLine("\nZad4_2")

  type Drzewo<'a> =
    | Puste
    | Wezel of 'a*Drzewo<'a>*Drzewo<'a>

  let rec Dodaj (v:'a, d: Drzewo<'a>) :Drzewo<'a> =
    match d with
```

```
    | Puste -> Wezel(v, Puste, Puste)
    | Wezel(wv, l, p) when wv <= v -> Wezel(wv, l ,Dodaj (v, p) )
    | Wezel(wv, l, p) -> Wezel(wv, Dodaj (v, l) , p )

let drzewo = Dodaj(5, Dodaj(10,Dodaj(2,Dodaj(3, Puste))))
Console.WriteLine($"{drzewo}")

/// ------- rozwiązanie
let rec agregacja (agr_fun, agr_val) = function
| Puste -> agr_val
| Wezel(v, l, p) ->
   let val_1 = (agr_fun agr_val v)
   let val_2 = agregacja (agr_fun, val_1) l
   agregacja (agr_fun, val_2) p
/// ------- rozwiązanie
let run = fun f -> agregacja (f, 0) drzewo

Console.WriteLine($"\t(+) -> {run(+)}")
Console.WriteLine($"\t(-) -> {run(-)}")
Console.WriteLine($"\t(/) -> {run(fun v wynik->v*2+wynik)}")

let wynik_string = agregacja ((fun a b -> $"{b} <-> {a}"), "") drzewo
Console.WriteLine($"\t(str) -> {wynik_string}")
```

Zadanie 4.4

Napisz funkcję, która przyjmuje listę F# i buduje z niej drzewo binarne.

```
let zadanie44() =
   Console.WriteLine("\nZad4_4")
   open Zad4_2
```

```
let list = [1;2;-5;4;9;-1]
Console.WriteLine($"{list}")

/// ------- rozwiązanie
let rec lista_na_drzewo (d:Drzewo<'a>) = function
    | [] -> d
    | v::n -> lista_na_drzewo (Dodaj ( v, d )) n
/// ------- rozwiązanie

let drzewko = lista_na_drzewo Puste list
Console.WriteLine($"{drzewko}")
```

Zadanie 4.6

Stwórz listę losowych liczb całkowitych zawierającą wartości z przedziału od –10 do 10. Następnie wykorzystaj funkcje z modułu List w celu podzielenia tej listy na dwie części: pierwszą zawierającą wartości dodatnie, drugą zawierającą wartości ujemne.
Z jakich funkcji możesz skorzystać, aby realizować to zadanie

```
printfn"Zadanie 4.6"
   let gll = new Random()
   let lista = List.init 100 (fun i->gll.Next(-10,10))
   let lista2 = List.filter(fun v-> v >= 0) lista
   let lista3 = List.filter(fun v-> v < 0)lista
   printfn"Lista gdzie liczby są >=0"
   for  i=0 to lista2.Length-1  do
            printf"%i. "(i+1)
            System.Console.Write((List.item i lista2).ToString()+"; ")
   printfn""
   printfn"Lista gdzie liczby są <0"
   for  i=0 to lista3.Length-1  do
            printf"%i. "(i+1)
```

```
        System.Console.Write((List.item i lista3).ToString()+"; ")
  let Lista=List.append lista2 lista3
  printfn""
  printfn"Lista po połaczeniu dwóch poprzednich"
  for  i=0 to Lista.Length-1  do
    printf"%i. "(1+i)
    System.Console.Write((List.item i Lista).ToString()+"; ")
  printfn""
```

Zadanie 4.7

Stwórz listę losowych liczb całkowitych czwierających wartości z przedziału od -10 do 10. Następnie wykorzystaj funkcje z modułu List w celu podzielenia tej listy na dwie części: pierwszą zawierającą wartości powyżej średniej, druga zawierającą wartości poniżej średniej. Z jakich funkcji możesz skorzystać aby zrealizować to zadanie

```
let Gll = new Random()
  let Lista_1= List.init 100 (fun i->Gll.Next(-10,10))
  let suma=List.fold(fun a b-> a+b)0 Lista_1
  let ilosc=Lista_1.Length
  let srednia:int=suma/ilosc
  printfn"Średnia: %i"srednia
  let lista_2 = List.filter(fun v-> v >= srednia) Lista_1
  let lista_3 = List.filter(fun v-> v < srednia)Lista_1
  let Lista1=List.append lista_2 lista_3
  for  i=0 to Lista1.Length-1  do
    printf"%i."(i+1)
    System.Console.Write((List.item i Lista1).ToString()+"; ")
```

Zadanie 4.8

Wczytaj plik “zad8.txt” Zawiera on zestaw współczynników dla równania kwadratowego. Podziel te parametry na trzy osobne listy w zależności czy

równanie opisane tymi parametrami ma zero, jedno rozwiązanie, dwa rozwiązania lub inne.

```
let zadanie48() =
   Console.WriteLine("\nZad4_8")
   open System.IO

   File.WriteAllText("Zad4_8.txt", "1 5 3\n4 3 1\n7 1 4\n8 7 0")



   let redFile = fun fnam ->
      let lines_to_points = fun (l:String) ->
         let l = l.Split(' ')
         (l.[0]|>float),(l.[1]|>float),(l.[2]|>float)
      Seq.toList( Seq.map lines_to_points (File.ReadLines fnam) )

   let parametry = redFile "Zad4_8.txt"
   Console.WriteLine($"parametry -> {parametry}")

   // axx+bx+c -> delta > 0 = dwa rozwiazania, delta = 0 jedno, delta < 0 - brak
   let delta = fun (a, b, c) -> b*b - 4.*a*c

   let dwa_rozwiazania = List.filter (fun abc -> delta abc > 0) parametry
   let jedno_rozwiazanie = List.filter (fun abc -> delta abc = 0) parametry
   let zero_rozwiazan = List.filter (fun abc -> delta abc < 0) parametry
   Console.WriteLine($"jedno -> {jedno_rozwiazanie}")
   Console.WriteLine($"dwa -> {dwa_rozwiazania}")
   Console.WriteLine($"zero -> {zero_rozwiazan}")
```

Zadanie 4.9

Wczytaj plik “zad9.txt”. W każdym wierszu zawiera on współrzędne punktu na płaszczyźnie dwuwymiarowej. Oblicz odległości pomiędzy dowolnymi

dwoma punktami, a następnie zapisz do pliku “TWOJA STARA.txt” pary tych punktów wraz z odległościami pomiędzy nimi posortowane rosnąco względem odległości

```
printfn"Zadanie 4.9"
  let zamienNaPunkt(ls:string[])={X=(float ls.[0]); Y=(float ls.[1])}

  let WczytajPunkt nazwaPliku=
    let linie= File.ReadLines nazwaPliku
    let lancuch= Seq.map (fun(l:string)->l.Split(' '))linie
    let punkty = Seq.map zamienNaPunkt lancuch
    punkty
  let odleglosc p1 p2 = sqrt( ((p2.X-p1.X)*(p2.X-p1.X)) + ((p2.Y-p1.Y)*(p2.Y-p1.Y)))

  let wczytajDane komunikat=
    printfn"%s"komunikat
    Console.ReadLine();

  let podajPunkt()=
    let x= float(wczytajDane"Podaj wspolrzedna x: ")
    let y= float(wczytajDane"Podaj wspolrzedna y: ")
    {X=x;Y=y}

  let Sprawdz punkt punkt1=
    printf"Punkt1::  %f  %f "punkt.X punkt.Y
    printf"Punkt2:: %f  %f "punkt1.X punkt1.Y
    printfn"odleglosc: %f"(odleglosc punkt punkt1)
    (odleglosc punkt punkt1)<>0.0

  let pokaz p =
    printfn""
```

```
let z=WczytajPunkt"C:\Users\Seba\Desktop\zad9.txt"
let punkt=podajPunkt()
let Odleglosc=Seq.filter(fun p->Sprawdz punkt p)z
Seq.iter pokaz Odleglosc
```

Zadanie 4.10

NApisz program podobny do przedstawionego w punkcie 1.8 tylko podziel punkty na dwie części. Osobne te które są wewnątrz okręgu oraz te które sa poza nim. Zapisz wyniki do dwóch osobnych plików

```
printfn"Zadanie 4.10"
  let zamienNaPunkt(ls:string[])=(float ls.[0], float ls.[1])

  let WczytajPunkt nazwaPliku=
    let linie= File.ReadLines nazwaPliku
    let lancuch= Seq.map (fun(l:string)->l.Split(' '))linie
    let punkty = Seq.map zamienNaPunkt lancuch
    punkty

  let wczytajWartosc komunikat=
    printfn"%s"komunikat
    System.Console.ReadLine()

  let wczytajOkrag()=
    let x= float(wczytajWartosc "Podaj wspolrzedna x: ")
    let y= float(wczytajWartosc "Podaj wspolrzedna y: ")
    let r= float(wczytajWartosc "Podaj promien: ")
    {srodek=(x,y);promien=r}

  let obliczOdleglosc(x1,y1) (x2,y2)=
    sqrt( (x1-x2)**2.0 + (y1-y2)**2.0 )
```

```
let wSrodku okrag punkt =
   (obliczOdleglosc okrag.srodek punkt)<=okrag.promien

let zaOkregiem okrag punkt =
      (obliczOdleglosc okrag.srodek punkt)>okrag.promien

let okrag=wczytajOkrag()
let punkty=WczytajPunkt"C:\Users\Seba\Desktop\zad9.txt"
let punktyWSrodku=
   Seq.filter (fun p -> wSrodku okrag p )punkty
use strumien = File.CreateText "C:\Users\Seba\Desktop\wSrodkuOkregu.txt"
let linie= Seq.map (fun p -> sprintf"%f %f"(fst p) (snd p))punktyWSrodku
Seq.iter (fun (l:string)-> strumien.WriteLine(l)) linie

let punktyZaOkregiem=
      Seq.filter (fun p -> zaOkregiem okrag p )punkty
use strumien = File.CreateText "C:\Users\Seba\Desktop\zaOkregiem.txt"
let linie= Seq.map (fun p -> sprintf"%f %f"(fst p) (snd p))punktyZaOkregiem
Seq.iter (fun (l:string)-> strumien.WriteLine(l)) linie
```

Zadanie 4.11

Wczytaj plik “iris.txt”. Opisuje on trzy gatunki kwiatów irysa: setosa, Velfaofaw, awfuwa gieuyg. Każdy kwiat jest opisany za pomocą czterech liczb określających odpowiednio długość :Listka kwiatu, szerokośc lista kwiatu, długośc płatka i szerokośc płatka.

```
printfn"Zadanie 4.11"
   let zamienNaKwiaty(l:string[])={DlugoscL=(float l.[0]);SzerokoscL=(float l.[1]);
                        DlugoscP=(float l.[2]);SzerokoscP=(float l.[3]); Gatunek=l.[4]}
   let WczytajKwiat nazwaPliku=
      let linie= File.ReadLines nazwaPliku
```

```
        let lancuch= Seq.map (fun(l:string)->l.Split(' '))linie
        let kwiaty = Seq.map zamienNaKwiaty lancuch
        kwiaty

   let pokazKwiaty kwiaty = printfn"%f %f :: %f %f ||| %s"kwiaty.DlugoscL kwiaty.SzerokoscL
kwiaty.DlugoscP kwiaty.SzerokoscP kwiaty.Gatunek

   let pomocnicza k=
      let dlugoscL=Seq.map(fun o ->{|DlugoscL=o.DlugoscL|})k
      let szerokoscL=Seq.map(fun o ->{|SzerokoscL=o.SzerokoscL|})k
      let dlugoscP=Seq.map(fun o ->{|DlugoscP=o.DlugoscP|})k
      let szerokoscP=Seq.map(fun o ->{|SzerokoscP=o.SzerokoscP|})k
      let DL_max=Seq.max dlugoscL
      let DL_min=Seq.min dlugoscL
      Console.WriteLine("Maksymalna dlugosc listka: "+DL_max.ToString()+"; Minimalna
dlugosc listka: "+DL_min.ToString())
      let SL_max=Seq.max szerokoscL
      let SL_min=Seq.min szerokoscL
      Console.WriteLine("Maksymalna szerokosc listka: "+SL_max.ToString()+"; Minimalna
szerokosc listka: "+SL_min.ToString())
      let DP_max=Seq.max dlugoscP
      let DP_min=Seq.min dlugoscP
      Console.WriteLine("Maksymalna dlugosc platka: "+DP_max.ToString()+"; Minimalna
dlugosc platka: "+DP_min.ToString())
      let SP_max=Seq.max szerokoscP
      let SP_min=Seq.min szerokoscP
      Console.WriteLine("Maksymalna szerokosc platka: "+SP_max.ToString()+"; Minimalna
szerokosc platka: "+SP_min.ToString())

   let kwiatek=WczytajKwiat"C:\Users\Seba\Desktop\iris.txt"
  //Seq.iter pokazKwiaty kwiatek
```

pomocnicza kwiatek

Zadanie 4.13

Stwórz listę losowych liczb całkowitych zawierającą wartości z przedziału od –10 do 10 (niech lista zawiera ok 1000 elementów). Określ ile razy każda z tych wartości wystapiła na liście. Wyniki przedstaw jako mapę, gdzie kluczem jest liczba od –10 do 10 a wartością liczba wystapień tej liczby.

```
let zadanie413() =
    Console.WriteLine("\nZad4_13")

    let rand = new System.Random()
    let rand_lsit = [for i in 1..1000 -> rand.Next(-10, 10)]
    Console.WriteLine($"rand {rand_lsit}")

    let count = fun v -> (List.filter (fun i -> i = v) rand_lsit).Length
    let map = Map.ofList([for i in -10..10->(i, count i)])
    Console.WriteLine($"map {map}")

//////////
```

Inaczej zrobione chyba

```
printfn"Zadanie 4.13"
    let pokazMape mapa= printfn("%A")mapa
    let gll = new Random()
    let lista = List.init 1000 (fun i->gll.Next(-10,10))
    for  i=0 to lista.Length-1  do
            printf"%i. "(i+1)
            System.Console.Write((List.item i lista).ToString()+"; ")
    let mutable Mten=0
    let mutable Mnine=0
    let mutable Meight=0
```

```
let mutable Mseven=0
let mutable Msix=0
let mutable Mfive=0
let mutable Mfour=0
let mutable Mthree=0
let mutable Mtwo=0
let mutable Mone=0
let mutable zero=0
let mutable ten=0
let mutable nine=0
let mutable eight=0
let mutable seven=0
let mutable six=0
let mutable five=0
let mutable four=0
let mutable three=0
let mutable two=0
let mutable one=0
let mutable inne=0

let x= List.toArray lista
for i=0 to x.Length-1 do
   match x.[i] with
   | -10-> Mten<- Mten+1
   | -9-> Mnine<-Mnine+1
   | -8-> Meight<-Meight+1
   | -7-> Mseven<-Mseven+1
   | -6-> Msix<-Msix+1
   | -5-> Mfive<-Mfive+1
```

```
    | -4-> Mfour<-Mfour+1
    | -3-> Mthree<-Mthree+1
    | -2-> Mtwo<-Mtwo+1
    | -1-> Mone<-Mone+1
    | 10-> ten<- ten+1
    | 9-> nine<-nine+1
    | 8-> eight<-eight+1
    | 7-> seven<-seven+1
    | 6-> six<-six+1
    | 5-> five<-five+1
    | 4-> four<-four+1
    | 3-> three<-three+1
    | 2-> two<-two+1
    | 1-> one<-one+1
    | 0-> zero<-zero+1
    |_-> inne<-0
  printfn""
  let mapa=Map.ofList[("-10",Mten);("-9",Mnine);("-8",Meight);("-7",Mseven);
                      ("-6",Msix);("-5",Mfive);("-4",Mfour);("-3",Mthree);("-2",Mtwo);("-
1",Mone);("0",zero);
                      ("10",ten);("9",nine);("8",eight);("7",seven);("6",six);
                      ("5",five);("4",four);("3",three);("2",two);("1",one);]
  printfn"Klucz Wartosc"
  Seq.iter pokazMape mapa
```

Zadanie 4.12

Wczytaj plik “iris.txt” Podziel dane na osobne grupy zgodnie z gatunkami kwiatu a następnie dla każdej grupy osobno wyznacz w każdej kolumnie wartość minimalną, maksymalną oraz średnią

```
printfn"Zadanie 4.12"
  let ZamienNaKwiaty(l:string[])={DlugoscL=(float l.[0]);SzerokoscL=(float l.[1]);
                                  DlugoscP=(float l.[2]);SzerokoscP=(float l.[3]); Gatunek=l.[4]}
```

```
let WczytajKwiat2 nazwaPliku=
    let linie= File.ReadLines nazwaPliku
    let lancuch= Seq.map (fun(l:string)->l.Split(' '))linie
    let kwiaty = Seq.map ZamienNaKwiaty lancuch
    kwiaty


let PokazKwiaty kwiaty = printfn"%f %f :: %f %f ||| %s"kwiaty.DlugoscL kwiaty.SzerokoscL
kwiaty.DlugoscP kwiaty.SzerokoscP kwiaty.Gatunek
let wyznaczSetosa k=
    let Setosa=Seq.filter(fun o ->o.Gatunek="Setosa")k
    //Seq.iter (fun o->printfn "%A" o)Setosa
    let dlugoscL=Seq.map(fun o ->{|DlugoscL=o.DlugoscL|})Setosa
    let szerokoscL=Seq.map(fun o ->{|SzerokoscL=o.SzerokoscL|})Setosa
    let dlugoscP=Seq.map(fun o ->{|DlugoscP=o.DlugoscP|})Setosa
    let szerokoscP=Seq.map(fun o ->{|SzerokoscP=o.SzerokoscP|})Setosa
    let DL_max=Seq.max dlugoscL
    let DL_min=Seq.min dlugoscL
    printfn"SETOSA"
    Console.WriteLine("Maksymalna dlugosc listka: "+DL_max.ToString()+"; Minimalna dlugosc
listka: "+DL_min.ToString())
    let SL_max=Seq.max szerokoscL
    let SL_min=Seq.min szerokoscL
    Console.WriteLine("Maksymalna szerokosc listka: "+SL_max.ToString()+"; Minimalna szerokosc
listka: "+SL_min.ToString())
    let DP_max=Seq.max dlugoscP
    let DP_min=Seq.min dlugoscP
    Console.WriteLine("Maksymalna dlugosc platka: "+DP_max.ToString()+"; Minimalna dlugosc
platka: "+DP_min.ToString())
    let SP_max=Seq.max szerokoscP
    let SP_min=Seq.min szerokoscP
    Console.WriteLine("Maksymalna szerokosc platka: "+SP_max.ToString()+"; Minimalna szerokosc
platka: "+SP_min.ToString())
let wyznaczVersicolour k=
```

```
        let Versicolour=Seq.filter(fun o ->o.Gatunek="Versicolour")k

        //Seq.iter (fun o->printfn "%A" o)Versicolour

        let dlugoscL=Seq.map(fun o ->{|DlugoscL=o.DlugoscL|})Versicolour

        let szerokoscL=Seq.map(fun o ->{|SzerokoscL=o.SzerokoscL|})Versicolour

        let dlugoscP=Seq.map(fun o ->{|DlugoscP=o.DlugoscP|})Versicolour

        let szerokoscP=Seq.map(fun o ->{|SzerokoscP=o.SzerokoscP|})Versicolour

        let DL_max=Seq.max dlugoscL

        let DL_min=Seq.min dlugoscL

        printfn"VERSICOLOUR"

        Console.WriteLine("Maksymalna dlugosc listka: "+DL_max.ToString()+"; Minimalna dlugosc
listka: "+DL_min.ToString())

        let SL_max=Seq.max szerokoscL

        let SL_min=Seq.min szerokoscL

        Console.WriteLine("Maksymalna szerokosc listka: "+SL_max.ToString()+"; Minimalna
szerokosc listka: "+SL_min.ToString())

        let DP_max=Seq.max dlugoscP

        let DP_min=Seq.min dlugoscP

        Console.WriteLine("Maksymalna dlugosc platka: "+DP_max.ToString()+"; Minimalna dlugosc
platka: "+DP_min.ToString())

        let SP_max=Seq.max szerokoscP

        let SP_min=Seq.min szerokoscP

        Console.WriteLine("Maksymalna szerokosc platka: "+SP_max.ToString()+"; Minimalna
szerokosc platka: "+SP_min.ToString())

    let wyznaczVirginica k=

        let Virginica=Seq.filter(fun o ->o.Gatunek="Virginica")k

        //Seq.iter (fun o->printfn "%A" o)Virginica

        let dlugoscL=Seq.map(fun o ->{|DlugoscL=o.DlugoscL|})Virginica

        let szerokoscL=Seq.map(fun o ->{|SzerokoscL=o.SzerokoscL|})Virginica

        let dlugoscP=Seq.map(fun o ->{|DlugoscP=o.DlugoscP|})Virginica

        let szerokoscP=Seq.map(fun o ->{|SzerokoscP=o.SzerokoscP|})Virginica

        let DL_max=Seq.max dlugoscL

        let DL_min=Seq.min dlugoscL
```

```
        printfn"Virginica"

        Console.WriteLine("Maksymalna dlugosc listka: "+DL_max.ToString()+"; Minimalna dlugosc
listka: "+DL_min.ToString())

        let SL_max=Seq.max szerokoscL

        let SL_min=Seq.min szerokoscL

        Console.WriteLine("Maksymalna szerokosc listka: "+SL_max.ToString()+"; Minimalna
szerokosc listka: "+SL_min.ToString())

        let DP_max=Seq.max dlugoscP

        let DP_min=Seq.min dlugoscP

        Console.WriteLine("Maksymalna dlugosc platka: "+DP_max.ToString()+"; Minimalna dlugosc
platka: "+DP_min.ToString())

        let SP_max=Seq.max szerokoscP

        let SP_min=Seq.min szerokoscP

        Console.WriteLine("Maksymalna szerokosc platka: "+SP_max.ToString()+"; Minimalna
szerokosc platka: "+SP_min.ToString())


    let kwiatki=WczytajKwiat2"C:\Users\Seba\Desktop\iris.txt"

    //Seq.iter PokazKwiaty kwiatki

    wyznaczSetosa kwiatki

    wyznaczVersicolour kwiatki

    wyznaczVirginica kwiatki
```

Lista5:

Zadanie 5.1

Napisz program z zadania 4.8 wykorzystując operator potokowy.

Wczytaj plik "zad8.txt" Zawiera on zestaw współczynników dla równania kwadratowego. Podziel te parametry na trzy osobne listy w zależności czy równanie opisane tymi parametrami ma zero, jedno rozwiązanie, dwa rozwiązania lub inne.

```
let zadanie51() =

    Console.WriteLine("\nZad5_1")

    open System.IO
```

```
File.WriteAllText("Zad4_8.txt", "1 5 3\n4 3 1\n7 1 4\n8 7 0")



let redFile = fun fnam ->
    // axx+bx+c -> delta > 0 = dwa rozwiazania, delta = 0 jedno, delta < 0 - brak
    let delta = fun (a, b, c) -> b*b - 4.*a*c


    fnam |> File.ReadLines
    |> Seq.map (fun l -> l.Split(' ') |> (fun ll -> (ll.[0]|>float),(ll.[1]|>float),(ll.[2]|>float)))
    |> Seq.toList
    |> fun list -> (
       List.filter (fun abc -> delta abc < 0) list,
       List.filter (fun abc -> delta abc = 0) list,
       List.filter (fun abc -> delta abc > 0) list)


"Zad4_8.txt"
|> redFile
|> fun (l1,l2,l3) -> Console.WriteLine($"brak -> {l1} \njedno -> {l2} \ndwa -> {l3} \n")
```

Zadanie 5.4

Napisz program, który wczyta od użytkownia z klawiatury liczbę całkowitą wartośc ta może być opcjonalna. Następnie podaną wartość należy podnieśc do kwadratu i wyświetlić na ekranie lub wyświetlić komunikat “Chuj ci w dupę” do rozwiązania zadania wykorzystaj typ Option i funkcje z modułu Option

```
let zadanie54() =
```

```
    Console.WriteLine("\nZad5_4")


    try (Console.ReadLine() |> int |> Some) with | _ -> None
    |> Option.map ( fun x -> $"{x*x}" )
    |> Option.defaultValue("Nie podałeś wartości")
    |> Console.WriteLine
```

Zadanie 5.6

```
let zadania56() =
    Console.WriteLine("\nZad5_6")


    let rec read = fun () ->
        "Podaj wartość:" |> Console.WriteLine
        Console.ReadLine() |> (fun l -> if l = "" then read() else l)


    try read() |> int |> (fun x -> Ok (x*x)) with | er -> Error er.Message
    |> fun x -> match x with | Error(ex) -> ex | Ok(v) -> v.ToString()
    |> Console.WriteLine
```

Zadanie 5.9

Napisz program, który wczyta od użytkownika z klawiatury liczbę całkowitą. Wartość ta może być opcjonalna. Uwzględnij fakt że może ona jednak być błędnie wprowadzona. Następnie podaną wartość należy podnieść do kwadratu i wyświetlić na ekranie lub wyświetlić odpowiednie komunikaty:

1. Jeżeli użytkownik nie podał wartości to: ”Nie podałeś wartości, to ja nie podam wyniku”.
2. Jeżeli użytkownik podał wartość ale błędną, to wyświetl komunikat o błędzie.

Do rozwiązania zadania wykorzystaj zarówno typ Result jak i typ Option.

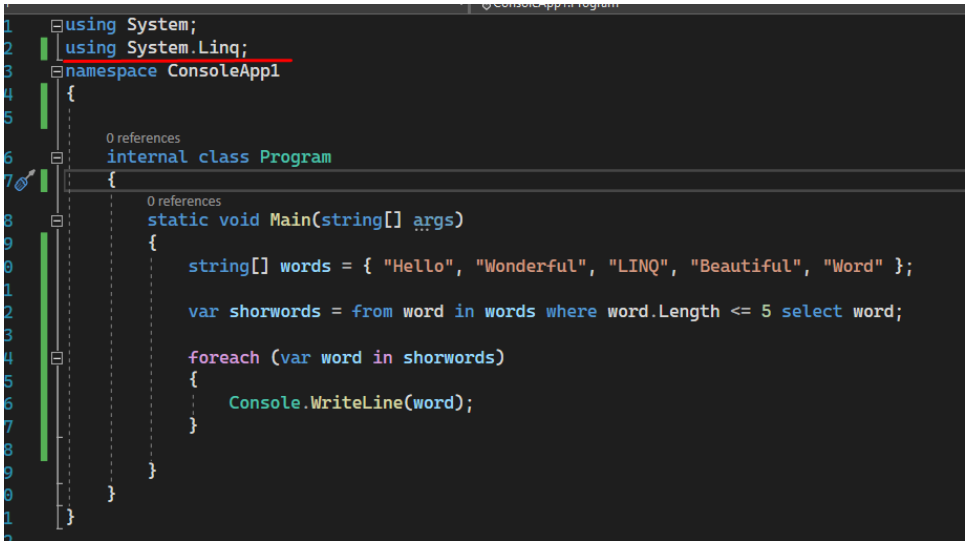
```
let zadanie59() =
    Console.WriteLine("\nZad5_9")


    Console.ReadLine()
    |> fun l ->  if l.Trim().Length = 0 then None else Some l
    |> Option.map (fun l -> (try (l |> int |> fun x -> Ok (x*x)) with | er -> Error er))
```

```
   |> fun x -> match x with None -> "Nie podałeś wartości" | Some r -> match r with |
Error(ex) -> ex.Message | Ok(v) -> v.ToString()
   |> Console.WriteLine
```

Zadanie 5.11

```
let zadanie511() =
   Console.WriteLine("\nZad5_11")

   let read_value = fun () -> Console.ReadLine().Trim()
   let parse_value = fun x -> try (x |> int |> Ok ) with | er -> Error er

   "podaj liczby:" |> Console.WriteLine

   let ResToString = function
   | Error s -> $"Error: {s}"
   | Ok s -> $"Wynik: {s}"

   Some (read_value ())
   |> Option.bind (fun s -> if s = "" then None else (parse_value s) |> Some )
   |> Option.bind (fun s -> (s, (read_value())) |> Some)
   |> Option.bind (fun (a, b) -> if b = "" then None else (a, parse_value b) |> Some )
   |> Option.map (fun ab ->
      match ab with
      | (Ok a, Ok b) -> Ok (a+b)
      | (Error a, _) -> Error $"Param1: {a.Message}"
      | (_, Error b) -> Error $"Param2: {b.Message}"
   )
   |> Option.defaultValue(Error "Nie podałeś wartości")
   |> ResToString |> Console.WriteLine
```

LINQ xd



```
using System;
using System.Linq;
namespace ConsoleApp1
{

    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string[] words = { "Hello", "Wonderful", "LINQ", "Beautiful", "Word" };

            var shorwords = from word in words where word.Length <= 5 select word;

            foreach (var word in shorwords)
            {
                Console.WriteLine(word);
            }

        }
    }
}
```