Observatii: Codul de mai jos a fost realizat si testat pe pagina online: https://www.tutorialspoint.com/compile_pascal_online.php

1. **Să se determine de câte ori apare cifra *c* în scrierea în baza *p* a numărului *n*.**

Program Cifre;

```
// descr: determina de cate ori apare o cifra intr-un numar in reprezentarea in baza p
// in: numarul, baza si cifra
// out: nr de aparitii
function cifreIterativ(numar, p, cifra:Longint):Longint;
var rez,ultimaCifra: Longint;
begin
    rez := 0;
    while numar > 0 do
    begin
        ultimaCifra := numar mod p;
        numar := numar div p;
        if ultimaCifra = cifra then
            rez := rez + 1;
    end;
    cifreIterativ := rez;
end;

// descr: determina de cate ori apare o cifra intr-un numar in reprezentarea in baza p
// in: numarul, cifra si baza
// out: nr de aparitii
function cifreRecursiv(numar, p, cifra:Longint):Longint;
var ultimaCifra:Longint;
begin
    if numar = 0 then
        cifreRecursiv := 0
    else
    begin
        ultimaCifra := numar mod p;
        numar := numar div p;
        if ultimaCifra = cifra then
            cifreRecursiv := 1 + cifreRecursiv(numar, p, cifra)
        else
            cifreRecursiv := cifreRecursiv (numar, p, cifra);
    end;
end;

// descr:  citeste un numar de la tastatura
// in: -
// out: nr citit
function citireNumar():Longint;
var x:Longint;
begin
    read(x);
    citireNumar :=x;
end;

// descr:  afiseaza un numar pe ecran
// in: numarul
// out: -
procedure afisareNumar(x:Longint);
begin
    writeln(x);
end;
```

```
//program principal pentru testare
var nr, p, c,rez:Longint;
begin
  nr := citireNumar();
  p := citireNumar();
  c := citireNumar();
  rez := cifreIterativ(nr, p, c);
  afisareNumar(rez);
end.
```

Exemple

| intrari | | | Iesiri |
|---|---|---|---|
| numar | baza | cif | rezultat |
| 6247891 | 8 | 2 | 3 |
| 6247891 | 8 | 3 | 1 |
| 6247891 | 8 | 4 | 0 |
| 6247891 | 16 | 5 | 3 |
| 6247891 | 16 | 15 | 1 |
| 6247891 | 2 | 1 | 15 |
| 6247891 | 2 | 0 | 8h |

2. **Fie f:[a,b] -> R o funcție continuă cu proprietatea *f(a)*f(b)* < 0. Scrieți un subalgoritm care determină o rădăcină a funcției *f* cu aproximarea epsilon.**

Program RadacinaFunctie;

```
// tipul "Functie" reprezinta orice functie care primeste ca parametru o valoare reala si
returneaza o
// valoare reala. Variabile de tipul "Functie" vor reprezenta functia din enunt.
type Functie = function(a: Real):Real;

//descr: determina o radacina a functie f, definite pe intervalul [a,b] cu aproximarea
epsilon
//in: a, b (capetele intervalului), f (functia) si epsilon (precizia)
//out: o radacina a functiei f cu precizia epsilon
function radacinaFunctieIterativ(a, b, epsilon: Real; f:Functie): Real;
var mijloc:Real;
begin
    while (b-a > epsilon) do
    begin
        mijloc := (a + b) / 2.0;
        if (f(a) * f(mijloc) < 0) then
            b := mijloc
        else
            a := mijloc;
    end;
    radacinaFunctieIterativ := (a + b)/2.0;
end;

//descr: determina o radacina a functie f, definite pe intervalul [a,b] cu aproximarea
//       epsilon
//in: a, b (capetele intervalului), f (functia) si epsilon (precizia)
//out: o radacina a functiei f cu precizia epsilon
function radacinaFunctieRecursiv(a, b, epsilon: Real; f:Functie):Real;
var mijloc:Real;
begin
    mijloc := (a + b) / 2.0;
    if (b - a < epsilon) then
        radacinaFunctieRecursiv := mijloc
    else if (f(a) * f(mijloc) < 0 ) then
        radacinaFunctieRecursiv := radacinaFunctieRecursiv(a, mijloc, epsilon, f)
    else
        radacinaFunctieRecursiv := radacinaFunctieRecursiv(mijloc, b, epsilon, f);
end;

//descr:  calculeaza valoarea functiei  x^3 - x – 2 intr-un punct
//in: x (punctul)
//out: valoarea functiei in punctul x
function functie1(x: Real):Real;
var aa:Real;
begin
    aa := x * x * x - x -2;
    functie1 := aa;
end;

//descr: calculeaza valoarea functiei  x^2+10x+3 intr-un punct
//in: x (punctul)
//out: valoarea functiei in punctul x
function functie2(x: Real):Real;
var aa:Real;
begin
    aa := x * x  + 10* x + 3;
    functie1 := aa;
end;
```

```
//descr:  citeste un numar real de la tastatura
//in: -
//out: nr citit
function citireNumar():Real;
var x:Real;
begin
    read(x);
    citireNumar := x;
end;

//descr:  afiseaza un numar pe ecran
//in: numarul
//out: -
procedure afisareNumar(x:Real);
begin
    writeln(x);
end;

//program principal pentru testare
var a,b,epsilon,rez:Real;
begin
  a := citireNumar();
  b := citireNumar();
  epsilon := citireNumar();
  rez := radacinaFunctieIterativ(a, b, epsilon, @functie1);
  afisareNumar(rez);
end.
```

Exemple

| Date de intrare | | | | Rezultat |
|---|---|---|---|---|
| a | b | eps | f | |
| -8 | 8 | 0.001 | X^3-x-2 | 1.521 |
| -8 | 8 | 0.00001 | X^3-x-2 | 1.52138 |
| -2 | 2 | 0.001 | X^2+10x+3 | -0.310059 |
| -2 | 2 | 0.00001 | X^2+10x+3 | -0.309582 |
| -3 | 5 | 0.001 | X^2+10x-3 | 0.291504 |
| -3 | 5 | 0.1 | X^2+10x-3 | 0.28125 |

3. **Fie șirul X = (1,2,2,3,3,3,4,5,5,5,5,5,6,7,7,7,7,7,7,7,8,9,10,11,...). Scrieți un subalgoritm care determină valoarea elementului de pe poziția *k*, fără a păstra șirul în memorie.**

```
Program ElementSir;

//desc: verifica daca numarul nr este prim
//in: nr
//out: true sau false
function prim(nr:Integer):Boolean;
var divizor:Integer; ePrim: Boolean;
begin
    ePrim := true;
    if nr <= 1 then
        ePrim := false
    else
    if (nr > 2) and (nr mod 2 = 0) then
        ePrim := false
    else
    begin
        divizor := 3;
        while divizor * divizor <= nr do
        begin
            if nr mod divizor = 0 then
                ePrim := false;
            divizor := divizor + 2;
        end;
    end;
    prim := ePrim;
end;

//descr: returneaza elementul k din sirul X
//in: k
//out: elementul de pe pozitia k din sir
function elemSir(k:Integer):Integer;
var poz_curent, val_curent, rezultat:Integer;
begin
    poz_curent := 0;
    val_curent := 1;
    rezultat := val_curent;

    while poz_curent < k do
    begin
        if prim(val_curent) then
        begin
            poz_curent := poz_curent + val_curent;
            rezultat := val_curent;
        end
        else
        begin
            poz_curent := poz_curent + 1;
            rezultat := val_curent;
        end;
        val_curent := val_curent + 1;
    end;
    elemSir := rezultat;
end;

//descr:  citeste un numar de la tastatura
//in: -
//out: nr citit
function citireNumar():Integer;
var x:Integer;
begin
    read(x);
    citireNumar := x;
end;
```

```
//descr:  afiseaza un numar pe ecran
//in: numarul
//out: -
procedure afisareNumar(x:Integer);
begin
    writeln(x);
end;

//program principal pentru testare
var k,rez:Integer;
begin
    k := citireNumar();
    rez := elemSir(k);
    afisareNumar(rez);
end.
```

Exemple

| Data de intrare | Rezultat |
|---|---|
| 5 | 3 |
| 10 | 5 |
| 20 | 7 |
| 50 | 15 |
| 101 | 23 |

4. **Fie șirul $X$ = (1,2,3,4,2,5,6,2,3,7,8,2,9,3,10,2,5,11,12,2,3,13,14,2,7,...). Să se afișeze elementele $X_n, X_{n+1},...X_{n+p}$, fără a păstra șirul X în memorie ($n, p > 0$)**

```
Program ElementSir;

//descr: verifica daca numarul nr este prim
//in: nr
//out: true sau false
function prim(nr:Integer):Boolean;
var divizor:Integer; ePrim: Boolean;
begin
    ePrim := true;
    if nr <= 1 then
        ePrim := false
    else
    if (nr > 2) and (nr mod 2 = 0) then
        ePrim := false
    else
        begin
            divizor := 3;
            while divizor * divizor <= nr do
            begin
                if nr mod divizor = 0 then
                    ePrim := false;
                divizor := divizor + 2;
            end;
        end;
    prim := ePrim;
end;

//descr: cauta urmatorul factor prim pentru nr, returneaza 0 daca nu mai sunt divizori
//in: nr, divizor_curent
//out: urmatorul factor prim  al lui nr, sau 0 daca nu mai exista factori primi
function urmFactorPrim(nr, div_curent:Integer):Integer;
var rez:Integer;
begin
    rez := 0;
    div_curent := div_curent + 1;
    while (div_curent < nr) and (rez = 0) do
    begin
        if (nr mod div_curent = 0) and prim(div_curent) then
            rez := div_curent;
        div_curent := div_curent + 1;
    end;
    urmFactorPrim := rez;
end;

//descr:  verifica daca pozitia curenta (pos_curent) trebuie afisata (este in intervalul
//        [n, n+p], si daca da, afiseaza valoarea val
//in: n, p, pos_curent, val
//out: -
procedure eInSolutie(n, p, poz_curent, val_curent:Integer);
begin
    if (poz_curent >= n) and (poz_curent <= (n+p)) then
        write(val_curent);
end;

//decr: verifica daca avand pozitia curenta (pos_curent) mai trebuie sa generam valori
//in: n, p, pos_curent
//out: true (daca nu mai trebuie sa generam numere), false (altfel)
function condOprire(n, p, poz_curent:Integer):Boolean;
begin
    if poz_curent <= (n+p) then
        condOprire := false
    else
        condOprire := true;
end;
```

```
//descr: genereaza elementele si le afiseaza pe cele din intervalul [n, n+p]
//in: n, p
//out: - (numerele sunt afisate pe ecran)
procedure generare(n, p:Integer);
var pos_curent, div_curent, val_curent:Integer;
begin
    pos_curent := 1;
    val_curent := 1;
    while condOprire(n, p, pos_curent) = false do
    begin
        eInSolutie(n, p, pos_curent, val_curent);
        pos_curent := pos_curent + 1;
        div_curent := urmFactorPrim(val_curent, 1);
        while (div_curent <> 0) and (condOprire(n, p, pos_curent) = false) do
        begin
            eInSolutie(n, p, pos_curent, div_curent);
            pos_curent := pos_curent + 1;
            div_curent := urmFactorPrim(val_curent, div_curent);
        end;
        val_curent := val_curent + 1;
    end;
end;

//descr:  citeste un numar de la tastatura
//in: -
//out: nr citit
function citireNumar():Integer;
var x:Integer;
begin
    read(x);
    citireNumar := x;
end;

//program principal pentru testare
var n,p:Integer;
begin
    n := citireNumar();
    p := citireNumar();
    generare(n,p);
end.
```

Exemple:

| Date de intrare | | Rezultat |
|---|---|---|
| n | p | |
| 5 | 5 | 2 5 6 2 3 7 |
| 10 | 7 | 7 8 2 9 3 10 2 5 |
| 15 | 5 | 10 2 5 11 12 2 |
| 15 | 10 | 10 2 5 11 12 2 3 13 14 2 7 |
| 101 | 11 | 46 2 23 47 48 2 3 49 7 50 2 5 |