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

```
#include <iostream>
using namespace std;

// descr: determina de cate ori apare o cifra intr-un numar in reprezentarea
// intr-o baza de numeratie
// in: numarul, cifra si baza
// out: nr de aparitii
int cifreIterativ(int numar, int p, int cifra) {
	int rezultat = 0;
	while (numar > 0) {
		int ultimaCifra = numar % p;
		if (ultimaCifra == cifra) {
			rezultat++;
		}
		numar = numar / p;
	}
	return rezultat;
}

// 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
int cifreRecursiv(int numar, int p, int cifra) {
	if (numar == 0) {
		return 0;
	}
	int ultimaCifra = numar % p;
	int numarNou = numar / p;
	if (ultimaCifra == cifra) {
		return 1 + cifreRecursiv(numarNou, p, cifra);
	}
	else {
		return cifreRecursiv(numarNou, p, cifra);
	}
}

// descr:  citeste un numar de la tastatura
// in: -
// out: nr citit
int citireNumar() {
	int x;
	cin >> x;
	return x;
}

// descr:  afiseaza un numar pe ecran
// in: numarul
// out: -
void afisareNumar(int x){
	cout << x;
}

// program principal pentru testare
int main(){
	numar = citireNumar();
	baza = citireNumar();
	cif = citireNumar();
	int rezultat = cifreIterativ(numar, baza, cifra);
	//int rezultat = cifreRecursiv(numar, baza, cifra);
	afisareNumar(rezultat);
}
```

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.**

```
#include <iostream>
using namespace std;

// 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.
typedef double(*functie)(double);

// 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
double radacinaFunctieIterativ(double a, double b, double epsilon, functie f) {
	//cat timp distanta dintre capetele intervalului este mai mare ca epsilon,
	//nu am ajuns la precizia ceruta
	while (b - a > epsilon) {
		double mijloc = (a + b) / 2.0;

		//trebuie sa vedem in care parte a intervalului [a, m) si [m, b] este radacina
		if (f(a) * f(mijloc) < 0) {
			b = mijloc;
		}
		else {
			a = mijloc;
		}
	}
	//returnam mijlocul intervalului [a,b]
	return (a + b) / 2.0;
}

// 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
double radacinaFunctieRecursiv(double a, double b, double epsilon, functie f) {

	double mijloc = (a + b) / 2.0;
	if (b - a < epsilon) {
		return mijloc;
	}
	else {
		if (f(a) * f(mijloc) < 0) {
			return radacinaFunctieRecursiv(a, mijloc, epsilon, f);
		}
		else {
			return radacinaFunctieRecursiv(mijloc, b, epsilon, f);
		}
	}
}

// descr:  calculeaza valoarea functiei  x^3 - x – 2 intr-un punct
// in: a (punctul)
// out: valoarea functiei in punctul a
double functie1(double a) {
	return a*a*a - a - 2;
}

// descr: calculeaza valoarea functiei  x^2+10x+3 intr-un punct
// in: a (punctul)
// out: valoarea functiei in punctul a
double functie2(double a) {
	return a*a + 10 * a + 3;
}
```

```
//descr:  citeste un numar real de la tastatura
//in: -
//out: nr citit
double citireNumar() {
	double x;
	cin >> x;
	return x;
}

// descr:  afiseaza un numar pe ecran
// in: numarul
// out: -
void afisareNumar(double x){
	cout << x;
}

// program principal pentru testare
int main() {
	double a = citesteNumar();
	double b = citesteNumar();
	double eps = citesteNumar();

	double rezultat = radacinaFunctieIterativ(a, b, eps, functie1);
	//double rezultat = radacinaFunctieRecursiv(a, b, eps, functie2);

	afiseazaNumar(rezultat);
}
```

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.

```
#include <iostream>
using namespace std;

// desc: verifica daca numarul nr este prim
// in: nr
// out: true sau false
bool prim(int nr) {
	bool prim = true;
	if (nr <= 1) {
		prim = false;
	}
	else if (nr > 2 && nr % 2 == 0) {
		prim = false;
	}
	else {
		int div = 3;
		while (div*div <= nr) {
			if (nr % div == 0) {
				prim = false;
			}
			div = div + 2;
		}
	}
	return prim;
}

// descr: returneaza elementul k din sirul X
// in: k
// out: elementul de pe pozitia k din sir
int elemSir(int k) {
	int poz_curent = 0;
	int val_curent = 1;
	int rezultat = val_curent;
	while (poz_curent < k) {
		if (prim(val_curent)) {
			//in sir numarul val_curent apare de val_curent ori, deci putem
			//sari peste atatea pozitii, care toate vor contine aceasta
			//valoare.
			poz_curent += val_curent;
			rezultat = val_curent;
		}
		else {
			//daca nu este prim, apare o singura data
			poz_curent++;
			rezultat = val_curent;
		}
		val_curent++;
	}
	return rezultat;
}


// descr:  citeste un numar de la tastatura
// in: -
// out: nr citit
int citireNumar() {
	int x;
	cin >> x;
	return x;
}
```

```
// descr:  afiseaza un numar pe ecran
// in: numarul
// out: -
void afisareNumar(int x){
	cout << x;
}

// program principal pentru testare
int main() {

	int k = citireNumar();
	int rez = elemSir(k);
	afisareNumar(rez);
	return 0;
}
```

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 ($n, p > 0$)

```
#include <iostream>
using namespace std;

// descr: verifica daca numarul nr este prim
// in: nr
// out: true sau false
bool prim(int nr) {
	bool prim = true;
	if (nr <= 1) {
		prim = false;
	}
	else if (nr > 2 && nr % 2 == 0) {
		prim = false;
	}
	else {
		int div = 3;
		while (div*div <= nr) {
			if (nr % div == 0) {
				prim = false;
			}
			div = div + 2;
		}
	}
	return prim;
}

// 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
int urmFactorPrim(int nr, int div_curent) {
	div_curent = div_curent + 1;
	while (div_curent < nr) {
		if (nr % div_curent == 0 && prim(div_curent)) {
			return div_curent;
		}
		div_curent = div_curent + 1;
	}
	return 0;
}

// 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: -
void eInSolutie(int n, int p, int pos_curent, int val) {
	if (pos_curent >= n && pos_curent <= n + p) {
		cout << val << " ";
	}
}

// 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)
bool condOprire(int n, int p, int pos_curent) {
	if (pos_curent > n + p) {
		return true;
	}
	else {
		return false;
	}
}

// descr: genereaza elementele si le afiseaza pe cele din intervalul [n, n+p]
// in: n, p
// out: - (numerele sunt afisate pe ecran)
```

```
void generare(int n, int p) {
	int pos_curent = 1;
	int val_curent = 1;
	while (!condOprire(n, p, pos_curent)) {
		eInSolutie(n, p, pos_curent, val_curent);
		pos_curent = pos_curent + 1;
		int factor = urmFactorPrim(val_curent, 1);
		while (factor != 0 && ! condOprire(n, p, pos_curent)) {
			eInSolutie(n, p, pos_curent, factor);
			pos_curent = pos_curent + 1;
			factor = urmFactorPrim(val_curent, factor);
		}
		val_curent = val_curent + 1;
	}
}

// descr:  citeste un numar de la tastatura
// in: -
// out: nr citit
int citireNumar() {
	int x;
	cin >> x;
	return x;
}

// program principal pentru testare
int main() {

	int n = citireNumar();
	int p = citireNuar();
	generare(n, p);
	return 0;
}
```

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