- Să se scrie funcţia pentru calcularea sumei elementelor unui masiv tridimensional. Să se folosească diferite variante pentru transmiterea masivului ca parametru.

Funcţia are ca parametri de intrare masivul tridimensional (*a*) şi dimensiunile sale efective (*m*, *n*, *p*). În prima variantă toate cele trei dimensiuni sînt precizate. În a doua variantă numărul de plane este omis (facilitate permisă în C). În a treia variantă se trimite un pointer spre o matrice. Cele trei variante de transmitere a masivului sînt echivalente.

```
int s1(int a[3][3][3],int m,int n,int p)
{ int s=0,i,j,k;
  for(i=0;i<m;i++)
    for(j=0;j<n;j++)
      for(k=0;k<p;k++)
        s+=a[i][j][k];
  return(s);}
int s2(int a[][3][3],int m,int n,int p)
{ int s=0,i,j,k;
  for(i=0;i<m;i++)
    for(j=0;j<n;j++)
      for(k=0;k<p;k++)
        s+=a[i][j][k];
  return(s);}
int s3(int (*a)[3][3],int m,int n,int p)
{ int s=0,i,j,k;
  for(i=0;i<m;i++)
    for(j=0;j<n;j++)
      for(k=0;k<p;k++)
        s+=a[i][j][k];
  return(s);}
```

- Să se scrie funcţia pentru afişarea conţinutului binar al unei zone de memorie în care se află memorat un şir de caractere.

Funcţia are ca parametru de intrare adresa şirului de caractere care trebuie afişat şi foloseşte o mască pentru a selecta fiecare bit al fiecărui caracter.

```
void bin(char *s)
{ unsigned char masca;
  while(*s)
   {masca=128;
    while(masca)
     {if(*s&masca)putch('1');
      else putch('0');
      masca>>=1;}
    s++;
    printf("\n");}}
```

- Să se scrie funcţia pentru aproximarea valorii soluţiei unei ecuaţii algebrice transcendente prin metoda bisecţiei.

Funcţia are ca parametri de intrare capetele intervalului în care se caută soluţia (*x0* şi *x1*), numărul maxim de iteraţii (*n*), precizia dorită (*eps*), funcţia asociată ecuaţiei (*f*) şi adresa unde se va înscrie soluţia. Prin numele funcţiei se returnează un cod de eroare cu următoarea semnificaţie: 0 – nu s-a găsit soluţie datorită numărului prea mic de iteraţii sau preciziei prea mari cerute; 1 – s-a obţinut soluţia exactă; 2 – s-a obţinut o soluţia aproximativă; 3 – intervalul dat nu conţine nici o soluţie.

-*varianta iterativă*

```
int bisectie(float x0,float x1,unsigned n,float eps,float
             (*f)(float), float *sol)
{ int cod=0;
  if ((*f)(x0)*(*f)(x1)>0) cod=3;
  else while((n)&&(!cod))
         {*sol=(x0+x1)/2;
          if((*f)(*sol)==0) cod=1;
          if(fabs(x0-x1)<=eps) cod=2;
```

```
          else {if((*f)(*sol)*(*f)(x0)<0)x1=*sol;
                else x0=*sol;
                n--;}}
  return cod;}
```

-varianta recursivă

```
int bisectie(float x0,float x1,unsigned n,float eps,float
             (*f)(float),float *sol)
{ int cod;
  if ((*f)(x0)*(*f)(x1)>0) cod=3;
  else if (n==0) cod=0;
       else {*sol=(x0+x1)/2;
             if((*f)(*sol)==0) cod=1;
             else if(fabs(x0-x1)<=eps) cod=2;
                  else {if((*f)(*sol)*(*f)(x0)<0)
                          cod=bisectie(x0,*sol,n-1,eps,f,sol);
                        else cod=bisectie(*sol,x1,n-1,eps,f,sol);}
  return cod;}
```

- Să se scrie funcţia pentru calculul lui *n!*, recursiv şi nerecursiv.
  Funcţia are ca parametru de intrare pe *n* şi returnează, prin numele ei, valoarea factorialului.

-varianta recursivă

```
long fact(long n)
{ long f;
  if (n==1)f=1;
  else f=n*fact(n-1);
  return(f);}
```

-varianta iterativă

```
long fact(long n)
{long f=1;
 for(long i=1;i<=n;i++)
   f*=i;
 return(f);}
```

- Să se scrie funcţia pentru calcularea termenului de ordin *n* al şirului Fibonacci, recursiv şi nerecursiv.

Funcţia are ca parametru de intrare indicele termenului pe care trebuie să îl calculeze şi returnează, prin numele ei, valoarea cerută. Indicii termenilor şirului încep de la 1.

-varianta iterativă

```
long fib(int n)
{ long f,a,b;
  int i;
  if ((n==1)||(n==2))f=1;
  else {a=1;b=1;
        for(i=3;i<=n;i++)
         {f=a+b;
          a=b;b=f;}
       }
  return(f);}
```

-varianta recursivă

```
long fib(int n)
{ long f;
  if ((n==1)||(n==2))f=1;
  else f=fib(n-1)+fib(n-2);
  return(f);}
```