- Să se scrie funcţia pentru calculul sumei a două polinoame.

Funcţia are ca parametri gradul primului polinom şi vectorul coeficienţilor săi, gradul celui de al doilea polinom şi vectorul coeficienţilor săi, vectorul în care se vor scrie coeficienţii polinomului rezultat şi adresa la care se va scrie gradul polinomului rezultat.

```
void s_polinom(int n,float a[],int m,float b[],float r[],int* p)
{ int i;
  *p=m>n?m:n;
  for(i=0;i<=*p;i++)
    r[i]=(i>n?0:a[i])+(i>m?0:b[i]);}
```

- Să se scrie funcţia pentru calcul produsului dintre două polinoame.

Funcţia are ca parametri gradul primului polinom, vectorul cu coeficienţii săi, gradul celui de al doilea polinom, vectorul cu coeficienţii săi, vectorul în care se vor scrie coeficienţii polinomului rezultat şi adresa la care se va scrie gradul polinomului rezultat.

```
void p_polinom(int n,float a[],int m,float b[],float r[],int* p)
{ int i,j,tt;
  float t[100];
  *p=0; tt=0;
  for(i=0;i<=n;i++)
   {tt=m+i;
    for(j=0;j<=m;j++)  t[j+i]=b[j]*a[i];
    for(j=0;j<i;j++) t[j]=0;
    s_polinom(tt,t,*p,r,r,p);
    getch();}
}
```

- Să se scrie funcţia pentru construirea unei noi matrice cu liniile şi coloanele unei matrice care nu conţin o anumită valoare.

Funcţia are ca parametri matricea iniţială şi dimensiunile sale (numărul linii-lor şi numărul coloanelor), noua matrice (adresa unde vor fi scrise elementele sale), adresele unde se vor scrie dimensiunile sale, numărul liniilor şi numărul de coloane-lor. Sînt apelate funcţiile *compactare* (exerciţiul 2.1.ix) şi *căutare* (exerciţiul 2.1.xx).

```
void nenule(float a[][20],int m,int n,float b[][20], int *p,int *q)
{ int i,j,k,r,s,l[20],c[20];
  k=0;
  r=0;
  for(i=0;i<m;i++)
    for(j=0;j<n;j++)
      if(!a[j][i])
       {l[k++]=i;
        c[r++]=j;}
  compactare(l,&k);
  compactare(c,&r);
  *p=0;*q=n;
  for(i=0;i<m;i++)
    if(cautare(l,k,i)==-1)
      {for(j=0;j<n;j++)
         b[*p][j]=a[i][j];
       (*p)++;}
  for(j=0;j<n;j++)
    if(cautare(c,r,j)!=-1)
      {for(s=j;s<n-1;s++)
         for(i=0;i<*p;i++)
           b[i][s]=b[i][s+1];
       n--;}
  *q=n;}
```

- Să se scrie funcţia pentru ridicarea la putere a unei matrice pătrate.

Funcţia are ca parametri matricea iniţială, dimensiunea ei, puterea la care se ridică şi matricea în care va scrie rezultatul. Sînt folosite funcţiile *copiere* (pentru copierea unei matrice în alta) şi *produs* (pentru înmulţirea a două matrice – exerciţiul 2.4.x).

```
void copiere(float a[][20],int m, float b[][20])
{ int i,j;
  for(i=0;i<m;i++)
    for(j=0;j<m;j++)
      b[i][j]=a[i][j];}

void putere(float a[][20],int m, int p,float b[][20])
{ int i,j,k;
  float c[20][20];
  for(i=0;i<m;i++)
    for(j=0;j<m;j++)
      c[i][j]=(i==j);
  for(i=0;i<p;i++)
   {produs(c,m,m,a,m,m,b,&m,&m);
    copiere(b,m,c);}
}
```

- Să se scrie funcţia pentru rezolvarea unui sistem algebric liniar de *n* ecuaţii cu *n* necunoscute, calculînd în acelaşi timp inversa matricei sistemului şi determinantul acesteia.

Funcţia are ca parametri matricea sistemului, gradul sistemului, vectorul termenilor liberi, limita sub care pivotul este considerat *0* (pivot 0 înseamnă o coloană nulă deci matrice neinversabilă şi sistem cu o infinitate de soluţii), matricea în care se va scrie inversa matricei sistemului şi vectorul în care se va scrie soluţia sistemului. Prin numele funcţiei se întoarce valoarea determinantului, care are şi rol de parametru de eroare (deteminant nul înseamnă matrice neinversabilă).

```
float inversa(float a[][20],int n,float b[],float eps,
              float inv[][20],float x[])
{ float c[20][20],e[20][20],d,aux;
  int i,j,k,p;
  d=1;                          //construire matrice de lucru
  for(i=0;i<n;i++)
   {for(j=0;j<n;j++)
     {c[i][j]=a[i][j]; c[i][j+n]=(i==j);}
    c[i][2*n]=b[i];}
  afisare(c,n,2*n+1);
  i=0;
  while((i<n)&&d)               //pivotare si calcul determinant
   {p=i;//cautare pivot pe coloana i
    for(j=i+1;j<n;j++)
      if(abs(c[j][i])>abs(c[p][i]))p=j;
    if(abs(c[p][i])<eps)d=0;    //aducere pivot in pozitie
    else{if(p!=i){aux=c[p][i];c[p][i]=c[i][i];c[i][i]=aux;d=-d;}
         d=d*c[i][i];
    for(j=0;j<n;j++) e[j][i]=0; //pivotare
    for(j=i;j<2*n+1;j++) e[i][j]=c[i][j]/c[i][i];
    for(j=0;j<n;j++)
      for(k=i;k<2*n+1;k++)
        if(j!=i)e[j][k]=(c[j][k]*c[p][i]-c[j][i]*c[i][k])/c[p][i];
    for(k=0;k<n;k++)
      for(j=i;j<2*n+1;j++)
        c[k][j]=e[k][j];}
    i++;}
  for(i=0;i<n;i++)              //separare rezultate
   {for(j=0;j<n;j++)
      inv[i][j]=c[i][j+n];
    x[i]=c[i][2*n];}
  return d;}
```