- Să se scrie funcţia pentru interclasarea elementelor a doi vectori sortaţi crescător.

Funcţia are ca parametri primul vector, numărul său de elemente, al doilea vector, numărul său de elemente, vectorul în care va scrie rezultatul şi adresa la care va scrie numărul de elemente din vectorul rezultat. Nu se întoarce nici un rezultat prin numele funcţiei.

```
void interclasare(float v[],int m,float w[],int n,float r[],int* p)
{int i,j;
 i=0;j=0;*p=0;
 while((i<m)&&(j<n))
   if(v[i]<w[j])
     r[(*p)++]=v[i++];
   else
     r[(*p)++]=w[j++];
 if(i==m) for(i=j;i<n;i++)
            r[(*p)++]=w[i];
 else     for(j=i;j<m;j++)
            r[(*p)++]=v[j];
}
```

- Să se scrie funcţia pentru calcularea produsului scalar dintre doi vectori.

Funcţia are ca parametri cei doi vectori şi numărul de elemente ale fiecăruia şi adresa unde va scrie parametrul de eroare. Prin numele funcţiei se întoarce produsul scalar. Parametrul de eroare are valoarea 0, dacă se calculează produsul, sau 1, dacă vectorii au lungimi diferite.

```
float prod_scal(float v[],int n,float v1[],int n1,int *er)
{ float p;
  int i;
  if(n1!=n)*er=1;
  else{*er=0;
       p=0;
       for(i=0;i<n;i++)
         p+=v[i]*v1[i];}
  return(p);}
```

- Să se scrie funcţia pentru calcularea produsului vectorial dintre doi vectori.

Funcţia are ca parametri cei doi vectori, numărul de elemente ale fiecăruia şi vectorul în care va scrie rezultatul. Prin numele funcţiei se întoarce parametrul de eroare. Parametrul de eroare are valoarea 0, dacă se calculează produsul, sau 1, dacă vectorii au lungimi diferite.

```
int prod_vect(float v[],int n,float v1[],int n1,float r[])
{ int i,er;
  if(n1!=n)er=1;
  else{er=0;
       for(i=0;i<n;i++)
       r[i]=v[i]*v1[i];}
  return(er);}
```

- Să se scrie funcţia pentru căutarea unui element într-un vector nesortat.

Funcţia are ca parametri vectorul, numărul de elemente şi valoarea căutată. Prin numele funcţiei se întoarce poziţia primei apariţii a elementului în vector sau –1 dacă elementul nu este găsit.

```
int cautare(float v[],int n,float x)
{ int i,er;
  er=-1;
  for(i=0;i<n;i++)
    if((v[i]==x)&&(er==-1)) er=i;
  return(er);}
```

- Să se scrie funcţia pentru căutarea unui element într-un vector sortat.

a) *Varianta iterativă*: funcţia are ca parametri vectorul, numărul de elemente şi valoarea căutată. Prin numele funcţiei se întoarce poziţia elementului găsit sau –1, dacă elementul nu a fost găsit.

```
int cautare_bin(float v[],int n,float x)
{ int i,j,er,p;
  er=-1;
  i=0;j=n-1;
  while((i<=j)&&(er==-1))
   {p=(i+j)/2;
```

```
    if(v[p]==x) er=p;
    else if(v[p]<x)i=p+1;
         else j=p-1;}
 return(er);}
```

b) *Varianta recursivă*: funcţia are ca parametri vectorul, capetele intervalului în care face căutarea (iniţial *0* şi *n-1*) şi valoarea căutată. Prin numele funcţiei se întoarce poziţia elementului găsit sau –1, dacă elementul nu a fost găsit.

```
int cautare_bin_r(float v[],int s,int d,float x)
{ int i,j,er,p;
  p=(s+d)/2;
  if(s>d)er=-1;
  else if(v[p]==x) er=p;
       else if(v[p]<x) er=cautare_bin_r(v,p+1,d,x);
            else       er=cautare_bin_r(v,s,p-1,x);
  return(er);}
```

- Să se scrie funcţia pentru determinarea numerelor naturale prime mai mici decît o valoare dată (maxim 1000) prin metoda ciurului lui Eratostene.

Funcţia are ca parametri limita maximă, vectorul în care va scrie numerele prime mai mici decît acea limită şi adresa unde va scrie numărul de numere găsite. Parametrul de eroare (1 dacă limita este mai mare de 1000, 0 dacă nu sînt erori) este întors prin numele funcţiei.

```
int eratostene(int x,int v[],int *y)
{ int i,er,q,v1[1000];
  if(x>500)er=1;
  else{er=0;
       for(i=0;i<x;i++) v1[i]=i;
         for(i=2;i<=sqrt(x);i++)
           {q=2*i;
            while(q<x)
              {v1[q]=0; q+=i;}
           }
       *y=0;
       for(i=0;i<x;i++)
         if(v1[i]) v[(*y)++]=v1[i];}
  return er;}
```

- Să se scrie funcţia pentru determinarea valorii unui polinom într-un punct dat.

Funcţia are ca parametri gradul polinomului *n*, vectorul coeficienţilor *a* (în ordine, primul coeficient fiind cel al termenului liber, în total *n+1* elemente) şi punctul în care se calculează valoarea polinomului. Prin numele funcţiei se întoarce valoarea calculată.

```
float polinom(int n,float a[],float x)
{ int i;
  float p;
  p=a[n];
  for(i=n;i>0;i--)
    p=p*x+a[i-1];
  return p;}
```