- Să se scrie funcţia pentru calculul sumei elementelor unei matrice folosind pointeri pentru adresare.

Funcţia are ca parametri adresa matricei şi dimensiunile ei. Prin numele funcţiei se întoarce suma elementelor matricei. Matricea a cărei adresă a fost primită este memorată în heap. Modificînd primul parametru al funcţiei în *float a[][20]*, se poate folosi funcţia pentru calculul sumei elementelor unei matrice statice cu 20 de coloane.

```
float suma(float **a,int m,int n)
{ int i,j; float s; s=0;
  for(i=0;i<m;i++)
    for(j=0;j<n;j++)
      s+=*(*(a+i)+j);
  return s;}
```

- Să se scrie funcţia pentru citirea unei matrice de la tastatură şi memorarea ei în heap.

Funcţia are ca parametri adresele unde va scrie dimensiunile matricei şi întoarce, prin numele ei, adresa matricei memorate în heap.

```
float ** citire(int *m,int *n)
{ int i,j;
  float **a;
  printf("nr linii: "); scanf("%d", m);
  printf("nr col  : "); scanf("%d", n);
  a=(float **)malloc(*m*sizeof(float*));
  for(i=0;i<*m;i++)
    a[i]=(float*)malloc(*n*sizeof(float));
  for(i=0;i<*m;i++)
    for(j=0;j<*n;j++)
     {printf("a[%d,%d]=",i,j);
      scanf("%f",&a[i][j]);}
  return a;}
```

- Să se scrie funcţia pentru citirea unui vector şi memorarea lui în heap.

Funcţia primeşte ca parametru adresa unde va scrie numărul elementelor vectorului şi returnează, prin numele ei, adresa vectorului memorat în heap.

```
float* citire(int* n)
{ int i;
  float* v;
  v=(float*)malloc(*n*sizeof(float));
  printf("n=");scanf("%d",n);
  for(i=0;i<*n;i++)
   {printf("v(%d)=",i);
    scanf("%f",&v[i]);}
  return v;}
```

- Să se scrie funcţia pentru sortarea unui vector folosind adresarea cu pointeri.

Funcţia are ca parametri adresa vectorului şi numărul său de elemente. Nu se întoarce nici o valoare prin numele funcţiei. Se poate apela atît pentru vectori memoraţi în heap cît şi pentru vectori memoraţi static.

```
void bule(float *v, int n)
{ int i,p; float a;
  p=1;
  while(p)
   {p=0;
    for(i=0;i<n-1;i++)
      if(*(v+i)>*(v+i+1))
       {a=*(v+i);
        *(v+i)=*(v+i+1);
        *(v+i+1)=a; p=1;}
   }
}
```

- Să se scrie subprogramul care verifică dacă o funcţie este injectivă.

O funcţie *f* este injectivă dacă pe a doua linie a matricei de reprezentare nu există dubluri.

Subprogramul are ca parametru funcţia şi întoarce valoarea 0 dacă funcţia nu este injectivă sau 1 dacă este injectivă.

```
int este_injectiva(RELATIE r)
{ int injectiva=1,i,j;
  for(i=0;(i<r.n-1)&&injectiva;i++)
    for(j=i+1;(j<r.n)&&injectiva;j++)
      if(r.r[1][i]==r.r[1][j])injectiva=0;
  return injectiva;}
```

- Să se scrie subprogramul care verifică dacă o funcţie este surjectivă.

În termenii reprezentării anterioare, o funcţie *f* este surjectivă dacă a doua linie a matricei de reprezentare conţine toate elementele mulţimii B.

Subprogramul are ca parametri funcţia şi mulţimea de valori. Prin numele subprogramului se întoarce valoarea 0 dacă relaţia nu este surjectivă sau 1 dacă este surjectivă.

```
int este_surjectiva(RELATIE r,MULTIME b)
{ int surjectiva=1,gasit,i,j;
  if(n>p) return 0;
  else{for(i=0;(i<b.n)&&surjectiva;i++)
        {for(j=0,gasit=0;(j<b.n)&&!gasit;j++)
           if(r.r[1][j]==b.x[i])gasit=1;
             if(!gasit)surjectiva=0;}
       return surjectiva;}
}
```