Laborator 3
1. Liste simplu înlănţuite
2. Liste dublu înlănţuite

1. Liste simplu înlănţuite

1. Să se creeze o lista cu elemente litere mici. Se cere:
a) afișarea listei
b) înlocuirea fiecărei vocale cu litera imediat următoare din alfabet și apoi sa se afiseze lista modificata
c) sa se transforme fiecare litera in litera mare și apoi sa se afiseze lista modificata.

```
#include <iostream>
using namespace std;
struct lista
{
char info;
lista *leg;
};

lista *p, *prim, *ultim;
int n;

void creare(lista *&prim, lista *&ultim)
{
   int i; char inf;
   cin>>n;
   cin>>inf;
   prim=new lista;
   prim->info=inf;
   prim->leg=NULL;
   ultim=prim;
   for(i=2;i<=n;i++)
   {
         cin>>inf;
         p=new lista;
         p->info=inf;
         p->leg=NULL;
         ultim->leg=p;
         ultim=p;
   }

}
```

```
void afisare(lista *prim)
{
   p=prim;
   while(p!=NULL)
   {
         cout<<p->info<<" ";
         p=p->leg;
   }
}
void inlocuire(lista *prim)
{
p=prim;
char voc[]="aeiou";

while(p!=NULL)
{
         for(int i=1;i<=5;i++)
         if(voc[i]==p->info)
                  p->info=p->info+1;
         p=p->leg;
}
cout<<endl<<"lista dupa inlocuire: ";
afisare(prim);
}
void transformare(lista *prim)
{
p=prim;
while(p!=NULL)
{
         p->info=p->info-32;
         p=p->leg;
}
cout<<endl<<"lista dupa transformare : ";
afisare(prim);
}
int main(void)
{
creare(prim,ultim);
cout<<endl<<"lista initiala : ";
afisare(prim);
inlocuire(prim);
transformare(prim);
return 0;
}
```

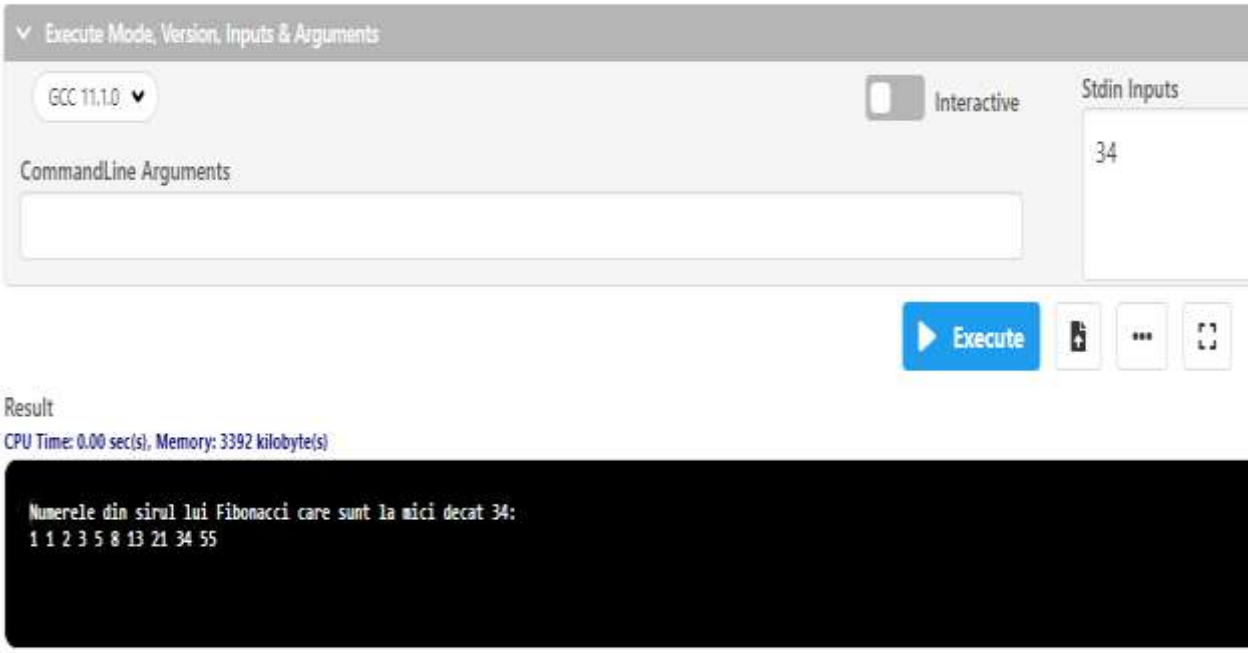
Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:

2. Să se creeze o lista cu primii nr termeni termeni din șirul lui Fibonacci. Se cere:
Exemplu: Pentru n=7 se obține lista cu elementele 1, 1, 2, 3, 5, 8, 13.

```
# include <iostream.h>
#include <iostream>
using namespace std;
struct lista{
int info;
lista *leg;
};

lista *p, *prim, *ultim;
int nr;

void creare(lista *&prim, lista *&ultim, int nr)
{
   int i,j,k,inf;
   prim=new lista;
   prim->info=1;
   prim->leg=NULL;
   p=new lista;
   p->info=1;
   p->leg=NULL;
   prim->leg=p;
   ultim=p;
   i=2;j=1;k=1;
   while(i <= nr){
         i=j+k;
         p=new lista;
         p->info=i;
         p->leg=NULL;
```

```
            ultim->leg=p;
            ultim=p;
            j=k;
            k=i;
    }
}
void afisare(lista *prim){
    p=prim;
    while(p!=NULL){
cout<<p->info<<" ";
p=p->leg;
    }
}
int main(void)
{
    //cout<<"Dati numarul pana la care se calculeaza termenii sirului lui Fibonacci ";
    cin>>nr;
creare(prim, ultim, nr);
cout<<"Numerele din sirul lui Fibonacci care sunt la mici decat "<<nr<<":\n";
afisare(prim);
return 0;
}
```

Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



Laborator 3

3. Se consideră o listă liniară simplu înlănțuită, alocată dinamic, în care elementele sunt de tipul declarat mai jos:

```
struct nod{
   int info;
   nod *leg;
};
```

în care câmpul **info** memorează un număr întreg, iar câmpul **leg** memorează adresa(legatura) următorului element al listei.

Să se scrie o funcție C++ cu următorul prototip: **int numarare(nod * p);**

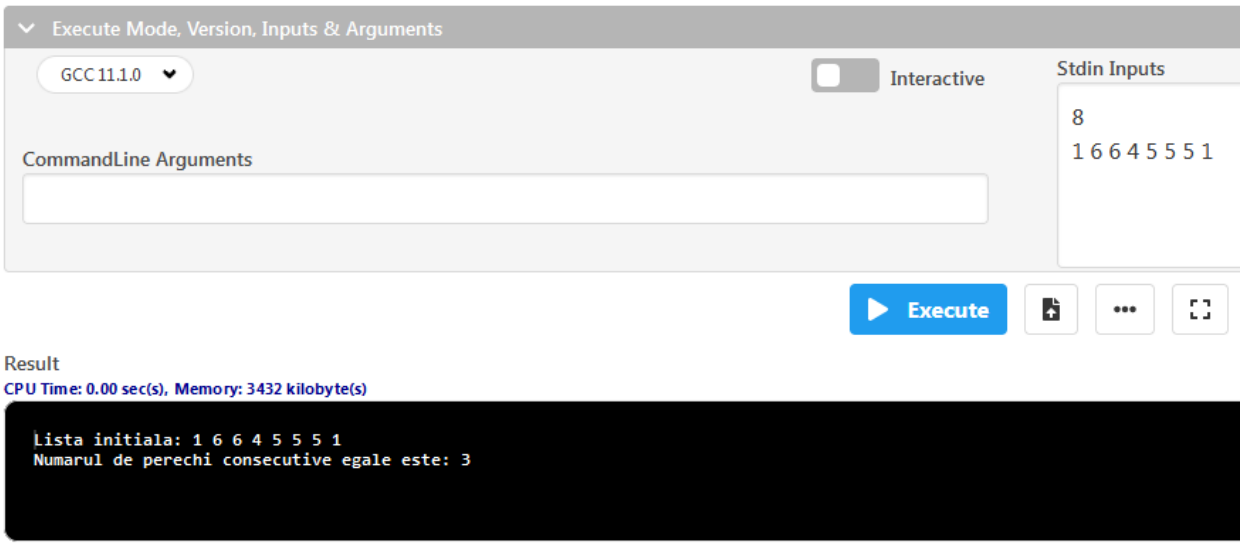
care determina și returnează numărul perechi de elemente consecutive egale din lista pentru care primul element are adresa memorată în pointerul p.

Exemplu: Dacă lista contine 8 valori: 1, 6, 6, 4, 5, 5, 5, 1, atunci funcția va returna valoarea 3.

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

struct nod{
int info;
nod *leg;
};

nod *p, *prim, *ultim;
int n;

void creare(nod *&prim, nod *&ultim, int n)
{
int i,j,k,inf;
prim = new nod;
cin>>prim->info;
prim->leg = NULL;
ultim = prim;
for(i=2; i <= n; i++)
{
        p = new nod;
        cin>>p->info;
        p->leg = NULL;
        ultim->leg = p;
        ultim = p;
}
}
void afisare(nod *prim)
{
p = prim;
while(p != NULL)
{
        cout<<p->info<<" ";
        p = p->leg;
}
```

```
}

int numarare(nod *p)
{
int k=0;
while(p->leg)
{
        if(p->info==p->leg->info)
        k++;
        p=p->leg;
}
return k;
}

int main(void)
{
cin>>n;
creare(prim, ultim, n);
cout<<"Lista initiala: ";
afisare(prim);
cout<<"\nNumarul de perechi consecutive egale este: "<<numarare(prim);
return 0;
}
```

Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



2. Liste dublu înlănţuite

1. Exemplu de utilizare a unei structuri de tip listă dublu înlănţuită împreună cu operaţiile ce se pot efectua asupra ei. Aplicaţia prezentată va afişa un meniu principal din care se poate selecta consecutiv operaţia dorită:

```
#include<iostream.h>
typedef struct lista
  {
   int inf;
   lista *as,*ad;
  };
lista *p,*prim,*ultim,*q;
int opt;
void creare(void)
{
int n;
prim=new lista;
ultim=new lista;
prim->ad=ultim;
prim->as=NULL;
ultim->as=prim;
cout<<"Dati informatia !=0 ";cin>>n;
while(n!=0)
{
        p=ultim;
        p->inf=n;
        ultim=new lista;
        p->ad=ultim;
        ultim->as=p;
        cout<<"Dati informatia !=0 ";cin>>n;
}
ultim->ad=NULL;
cout<<"Apasa o tasta pentru continuare"<<endl;
getch();
}
void afisare_stanga_dreapta(void)
{
  cout<<"parcurgerea listei de la stanga la
dreapta"<<endl;
  p=prim->ad;
  if(p==ultim) cout<<"Lista este VIDA!!";
  else
  {
do{
        cout<<p->inf<<" ";
```

```
        p=p->ad;
}while(p!=ultim);
  }
  cout<<"Apasa o tasta pentru continuare"<<endl;
  getch();
}
void afisare_dreapta_stanga(void)
{
  cout<<"parcurgerea listei de la dreapta la stanga"<<endl;
  p=ultim->as;
  if(p==prim) cout<<"Lista este VIDA!!";
  else
  {
do{
        cout<<p->inf<<" ";
        p=p->as;
}while(p!=prim);
  }
  cout<<"Apasa o tasta pentru continuare"<<endl;
  getch();
}
void adaug_la_inceput(void)
{
int n;
cout<<"Dati informatia care se adauga ";cin>>n;
p=new lista;
prim->inf=n;
prim->as=p;
p->ad=prim;
prim=p;
prim->as=NULL;
cout<<"Apasa o tasta pentru continuare"<<endl;
getch();
}
void adaug_la_sfarsit()
{
int n;
cout<<"Dati informatia care se adauga ";cin>>n;
p=new lista;
ultim->inf=n;
ultim->ad=p;
p->as=ultim;
ultim=p;
ultim->ad=NULL;
cout<<"Apasa o tasta pentru continuare"<<endl;
getch();
}
void stergere(void)
```

```
{
int n;
p=prim;
cout<<"Dati informatia care se va sterge  ";cin>>n;
if(p->inf!=n)
{
        q=p->ad;
        while(q->inf!=n)
        {
                p=q;
                q=q->ad;
        }
        q->ad->as=p;
        p->ad=q->ad;
}
else
{
   prim=p->ad;
   prim->as=NULL;
}
cout<<"Apasa o tasta pentru continuare"<<endl;
getch();
}

int main(void)
{
 do{
     cout<<endl;
     cout<<"   1.  Crearea listei"<<endl;
     cout<<"   2.  Adaugare la inceputul listei"<<endl;
     cout<<"   3.  Adaugare la sfarsitul listei"<<endl;
     cout<<"   4.  Parcurgerea de la stanga la dreapta"<<endl;
     cout<<"   5.  Parcurgerea de la dreapta la stanga"<<endl;
     cout<<"   6.  Stergerea unei informatii date"<<endl;
     cout<<"   7.  Terminare program"<<endl;
     cout<<endl;
     cout<<"Dati optiunea ";cin>>opt;
     switch (opt){
         case 1:{creare();break;}
         case 2:{adaug_la_inceput();break;}
         case 3:{adaug_la_sfarsit();break;}
         case 4:{afisare_stanga_dreapta();break;}
         case 5:{afisare_dreapta_stanga();break;}
         case 6:{stergere();break;}
 }
  }while(opt!=7);
}
```

Laborator 3

2. Sa se construiasca o lista care sa poata fi parcursa in ambele sensuri, apoi sa se elimine din lista primul si ultimul element egal cu un numar a dat.

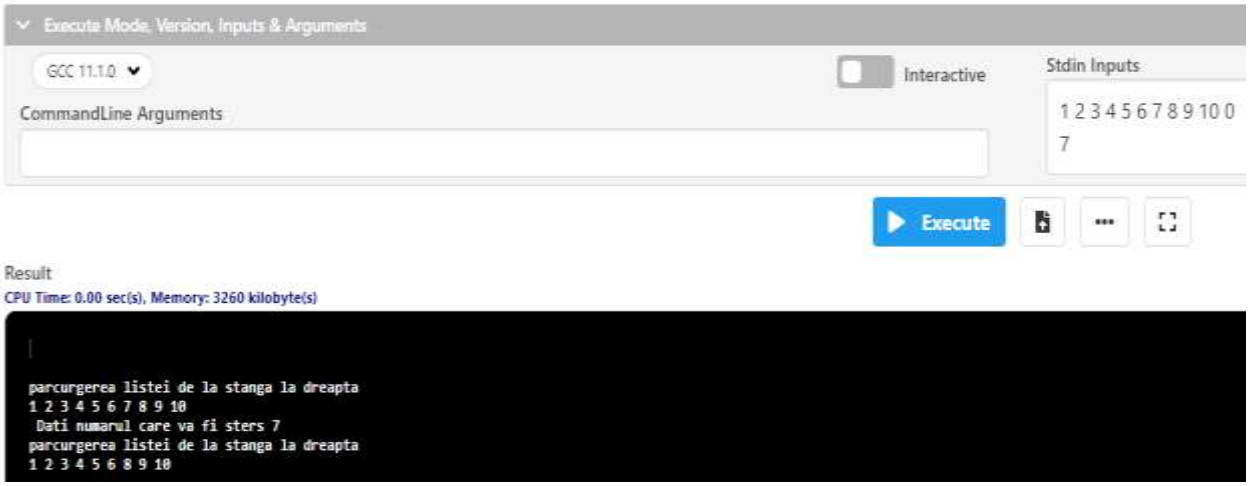
Vom folosi variabilele **prim** - adresa primului elem. din lista
ultim - adresa ultimului elem. din lista
p - pointer curent (de lucru)

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

struct lista
      {
       int inf;
       lista *as,*ad;
      };
lista *p,*prim,*ultim,*q;
int a;
void creare(void)
{
   int n;
   prim=new lista;
   ultim=new lista;
   prim->ad=ultim;
   prim->as=NULL;
   ultim->as=prim;
   //cout<<"Dati informatia !=0 ";
   cin>>n;
   while(n!=0)
   {
      p=ultim;
      p->inf=n;
      ultim=new lista;
      p->ad=ultim;
      ultim->as=p;
      //cout<<"Dati informatia !=0 ";
      cin>>n;
   }
   ultim->ad=NULL;
}
void cautare_stanga_dreapta(void)
{
   p=prim;
   q=p->ad;
   while(p!=NULL)
   {
      while(q->inf!=a)
      {
         p=q;
```

```
            q=q->ad;
        }
        if(q->inf==a)
        {
            q->ad->as=p;
            p->ad=q->ad;
            break;
        }
        p=p->ad;
    }
}
void cautare_dreapta_stanga(void)
{
    p=ultim;
    q=p->as;
    while(p!=NULL)
    {
        while(q->inf!=a)
        {
            p=q;
            q=q->as;
        }
        if(q->inf==a)
        {
            q->as->ad=p;
            p->as=q->as;
            break;
        }
        p=p->as;
    }
}
void afisare_stanga_dreapta(void)
{
   cout<<"\nparcurgerea listei de la stanga la dreapta"<<endl;
   p=prim->ad;
   if(p==ultim) cout<<"Lista este VIDA!!";
   else
   {
    do{
        cout<<p->inf<<" ";
        p=p->ad;
    }while(p!=ultim);
   }
}

int main(void)
{
     cout<<endl;
```

```
    creare();
    afisare_stanga_dreapta();
    cout<<endl;
    cout<<" Dati numarul care va fi sters ";
    cin>>a;cout<<a;
    cautare_stanga_dreapta();
    afisare_stanga_dreapta();
    cout<<endl;
    cautare_dreapta_stanga();
    afisare_stanga_dreapta();
    return 0;
}
```

Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



3. Se consideră o listă liniară dublu înlănțuită, alocată dinamic, în care elementele sunt de tipul declarat mai jos:

```
struct nod{
   int info;
   nod * ant,*urm;
};
```

în care câmpul info memorează un număr întreg, câmpul ant memorează adresa elementului anterior al listei, iar campul urm memorează adresa elementului următor al listei.
Să se scrie o funcție C++ cu următorul prototip:

void StergeAfterQ(nod*&prim,nod*&ultim, nod*q);

care sterge nodul de dupa nodul de adresa q al listei pentru care primul element are adresa memorată în pointerul prim si ultimul element are adresa memorata in pointerul ultim.

```
#include <iostream>
```

```
using namespace std;
int n, a;
struct nod{
   int info;
   nod * ant,*urm;
};
nod *prim, *ultim, *p;

void AdaugareFinal(nod * & prim , nod * & ultim, int x)
{
   if(prim == NULL)
   {
      prim = ultim = new nod;
      prim->info = x;
      prim->ant = prim->urm = NULL;
   }
   else
   {
      nod *p = new nod;
      p->info = x;
      p->urm = NULL;
      p->ant = ultim;
      ultim->urm = p;
      ultim = p;
   }
}

void StergeAfterQ(nod * & prim, nod * & ultim, nod * q)
{
   if(q != ultim)
   {
      nod *p;
      p = q->urm;

      q->urm = p->urm;
      p->urm->ant = q;

      delete p;
   }
}

void Afisare (nod *prim)
{
   nod *p;
   p = prim;
   cout<<"\nLista are urmatoarele elemente: ";
   while(p)
   {
```

```
        cout<<p->info<<" ";
        p = p -> urm;
    }
}
int main()
{
    cin>>n;
    while(n != 0)
    {
        AdaugareFinal(prim, ultim, n);
        cin>>n;
    }
    Afisare(prim);
    //cout<<"\nDati informatia dupa care se elimina elementul din lista ";
    cin>>a;
    p = prim;
    while( p != NULL )
    {
        if(p->info == a)
            StergeAfterQ(prim, ultim, p);
        p = p -> urm;
    }
    Afisare(prim);
    return 0;
}
```

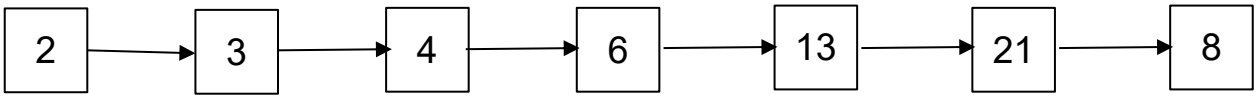
Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:

Probleme propuse spre rezolvare

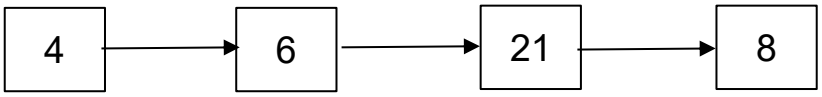
Laborator 3

1. Liste liniare simplu inlantuite

1. Eliminarea elementelor prime dintr-o lista.
Exemplu:
Lista inițial conține următoarele valori:



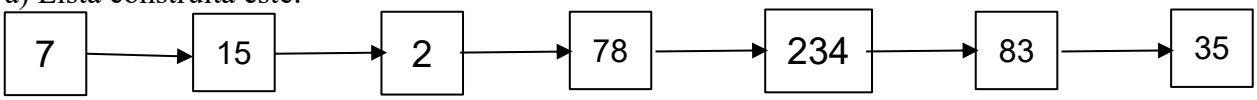
Lista finala, după eliminarea valorilor:



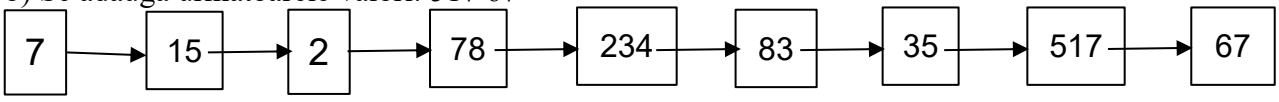
2. Să se creeze o listă liniară simplu înlănțuită care prelucrează n numere întregi.

a) Sa se afișeze conținutul listei
b) Sa se mai adauge 2 elemente in lista si sa se afiseze
c) Sa se mai adauge m elemente in lista si sa se afiseze
d) Sa se numere cate numere prime sunt in lista
e) Sa se determine suma elementelor din lista
f) Sa se determine daca lista este ordonata crescator
g) Sa se determine minimul (maximul) listei
h) Sa se mai creeze o lista care se va adăuga la sfarsitul listei (concatenare). Sa se afiseze lista nouă astfel generata
i) Se vor dubla elementele pentru care suma cifrelor este mai mica decat 10

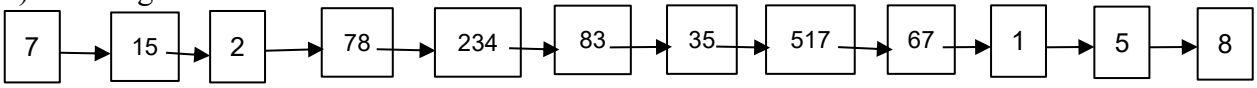
Exemplu:
Pentru n = 7 și următoarele valori: 7 15 2 78 234 83 35
a) Lista construita este:



b) Se adăuga următoarele valori: 517 67



c) Se adauga m=3 valori: 1 5 8:



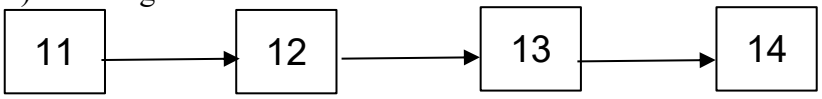
d) In lista sunt 5 numere prime: 7, 2, 5, 67, 83

e) Suma elementelor din lista este: 1052

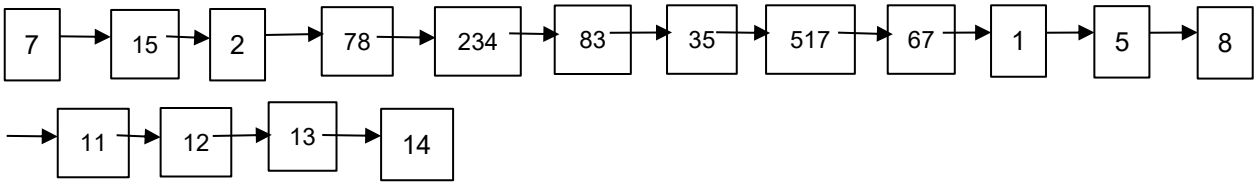
f) Lista nu este ordonata crescator

g) Minimul din lista este = 1, iar maximul din lista este = 517

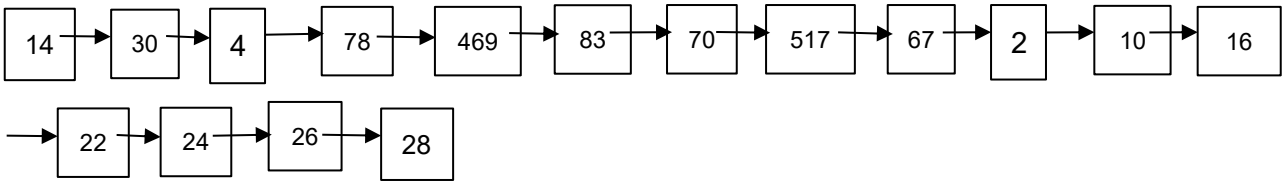
h) Se adaugă următoarea listă:



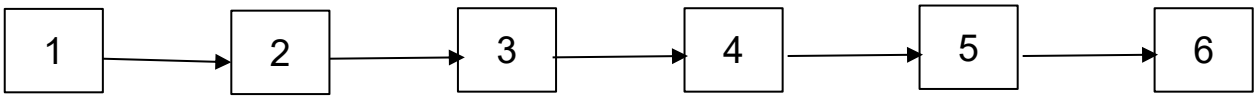
Se obtine lista concatenata:



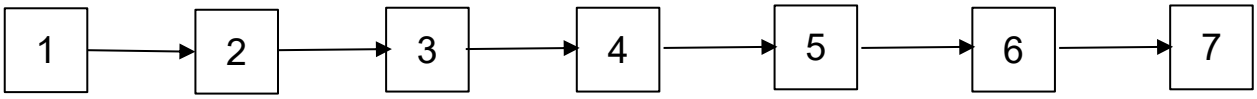
i) Lista obtinuta dupa dublarea celor pentru care suma cifrelor este mai mica decat 10 este:



3. Sa se formeze o lista care contine elementele unei matrici.
Exemplu:
Pentru numărul de linii n = 2 și numărul de coloane m = 3 și următoarele valori din matrice:
1 2 3
4 5 6
Lista obtinuta cu elementele matricei:

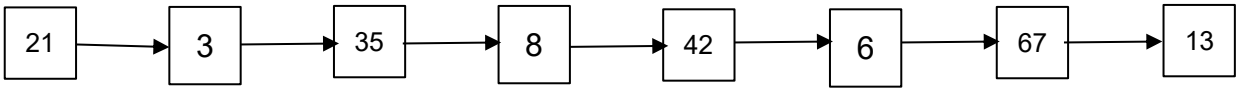


4. Sa se formeze o listă care conține elementele unui vector
Exemplu:
Pentru numărul de elemente n = 7 și valorile din vector: 1 2 3 4 5 6 7

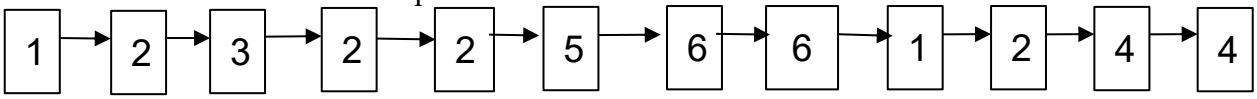


5. Sa se genereze o lista care va reține numere întregi și după fiecare numar suma cifrelor sale
Exemplu:
Se vor introduce următoarele valori:

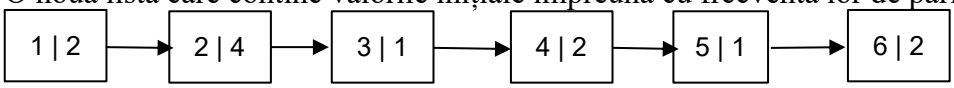
Se va obține lista cu valorile următoare:



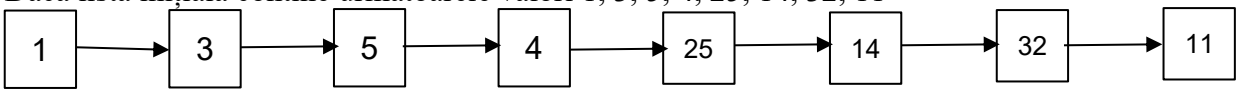
6. Dată fiind o lista cu elemente numere întregi, sa se formeze o a doua listă care conține elementele distincte din lista data, împreuna cu frecvența lor de aparitie.
Exemplu:
Lista initiala cu valori care se repeta:



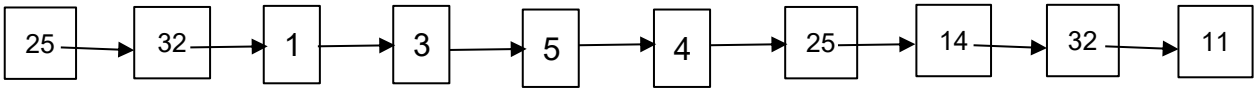
O noua lista care contine valorile inițiale impreuna cu frecventa lor de paritie in lista intiala:



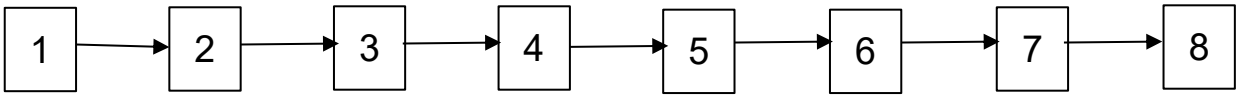
7. Fie o lista de numere întregi distincte. Primele două valori (cele mai mari) se vor insera la începutul listei.
Exemplu:
Dacă lista inițială contine următoarele valori 1, 3, 5, 4, 25, 14, 32, 11



Atunci se va obține următoarea listă: 25, 32, 1, 3, 5, 4, 25, 14, 32, 11



8. Sa se afiseze elementele (elementul) din mijlocul unei liste de întregi
Exemplu:
Lista inițială contine următoarele valori:



Elementele aflate în mijlocul listei sunt: 4 si 5.

9. O lista contine titlul și autorul unei cărți, precum și editura. Sa se afiseze acele carti din lista editate la editura Academica Brancusi din Targu-Jiu.

10. Să se creeze o lista care prelucrează numere complexe. Lista se citește de pe fiecare linie cate doua valori reprezentand partea reala si respective cea zecimala

a) Sa se afișeze conținutul listei
b) Sa se determine numărul complex cu modulul cel mai mare
c) Sa se determine suma (produsul) elementelor din lista

2. Liste liniare dublu inlantuite

1. Să se ruleze programele prezentate mai sus, urmărind apelurile şi valorile parametrilor de apel.

2. Există numere prime mai mici de 100.000 care au suma cifrelor egală cu 13 ? Dacă da, atunci să se formeze o listă circulară cu aceste numere, în ordinea găsirii lor.

3. Într-o listă dublu înlănţuită de n numere reale deja construită, pentru un element x al listei, să se adauge în faţa lui media aritmetică a elementelor aflate înaintea lui, iar după el să se adauge media geometrică a elementelor aflate după el în lista iniţială.

4. Se considera o lista liniara dublu inlantuita ale cărei noduri sunt memorare cifre. Sa se scrie o funcție care primește ca parametru adresa primului nod al listei și verifică dacă numărul care se compune din cifrele memorare în lista în ordine este sau nu palindrom. Funcția va returna 1 dacă este palindrom și 0 în caz contrar.

5. Se considera o lista liniara dublu inlantuita. Sa se scrie o funcție care primeste ca parametru adresa primului nod al listei și muta ultimul nod in fata primului.