Laborator 4
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);
```

```
}
```

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.h>
typedef 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;
        cout<<"Apasa o tasta pentru continuare"<<endl;
        getch();
}
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;
        }
        cout<<"Apasa o tasta pt. continuare"<<endl;
        getch();
}
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;
        }
        cout<<"Apasa o tasta pt. 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();
}

int main(void)
{
          cout<<endl;
          creare();
          afisare_stanga_dreapta();
          cout<<endl;
          cout<<" Dati numarul care va fi sters ";cin>>a;
          cautare_stanga_dreapta();
          afisare_stanga_dreapta();
          cout<<endl;
          cautare_dreapta_stanga();
          afisare_stanga_dreapta();
}
```

3. Sa se construiasca o lista care sa poata fi parcursa in ambele sensuri, apoi sa se elimine din lista elementele care se repeta.

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

```
#include<iostream.h>
typedef struct lista
          {
            int inf;
            lista *as,*ad;
          };
lista *p,*prim,*ultim;
void creare(void)
{
        int n;
        prim=NULL;
        cout<<"Dati informatia !=0 ";cin>>n;
        while(n!=0)
        {
                if(prim==NULL)
                {
                        prim=new lista;
                        prim->inf=n;
```

```
                        prim->as=NULL;
                        prim->ad=NULL;
                        ultim=prim;
                }
                else
                {
                        p=new lista;
                        p->inf=n;
                        p->ad=NULL;
                        p->as=ultim;
                        ultim->ad=p;
                        ultim=p;
                }
                cout<<"Dati informatia !=0 ";cin>>n;
        }
        cout<<"Apasa o tasta pentru continuare"<<endl;
        getch();
}
void afisare_stanga_dreapta(void)
{
  cout<<"parcurgerea listei de la stanga la dreapta"<<endl;
  p=prim;
  do{
        cout<<p->inf<<" ";
        p=p->ad;
  }while(p!=NULL);
  cout<<"Apasa o tasta pentru continuare"<<endl;
  getch();
}
void eliminare(void)
{
        lista *p,*q,*t;
        int k;
        p=prim;
        while(p!=NULL)
        {
            k=0;
            q=p->ad;t=q->as;
            while( (q!=NULL) && (k==0) )
            {
                if(q->inf==p->inf) k=1;
                else {t=q;q=q->ad;}
            }
            if(k==1)
            {
                t->ad=q->ad;
                q->ad->as=t;
            }
```

```
            p=p->ad;
        }
}
int main(void)
{
        cout<<endl;
        creare();
        afisare_stanga_dreapta();
        cout<<endl;
        cout<<endl;
        eliminare();
        afisare_stanga_dreapta();
}
```

4. Sa se creeze o lista liniara dublu inlantuita cu noduri care sa contina:
- campul info numar intreg;
- campurile ant si urm care sa contina informatii de legatura spre nodul anterior, respectiv urmator din lista.
a) Sa se afiseze lista in ambele sensuri
b) Sa se stearga toate nodurile din lista cu exceptia primului si a ultimului nod.

```
#include<fstream.h>

ifstream f("date.in");
ofstream g("date.out");

typedef struct nod{
        int info;
        nod *ant,*urm;
}LISTA;
LISTA *prim;

void adaugf(LISTA *&prim, int x)
 {
        LISTA *nou=new LISTA;
        nou->info=x;
        nou->urm=prim;
        nou->ant=0;
        prim->ant=nou;
        prim=nou;
 }

void creare(LISTA *&prim)
 {
        int x;
        while(f>>x) adaugf(prim,x);
```

```
}

void afis(LISTA *prim)
{
        LISTA *p=prim;
        while(p->urm)
        {
        g<<p->info<<" ";
        p=p->urm;
        }
        g<<p->info<<endl;
        while(p)
        {
        g<<p->info<<" ";
        p=p->ant;
        }
        g<<endl;
}

void sterg(LISTA *&p)
{
        LISTA *q,*r;
        q=p->ant;
        r=p->urm;
        q->urm=r;
        r->ant=q;
        delete p;
}

int main()
{
        creare(prim);
        afis(prim);
        LISTA *p=prim->urm;
        while(p->urm)
        {
              LISTA *q=p;
              p=p->urm;
              sterg(q);
        }
        afis(prim);
}
```

Probleme propuse spre rezolvare

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 carei noduri sunt memorare cifre. Sa se scrie o functie care primeste ca parametru adresa primului nod al listei si verifica daca numarul care se compune din cifrele memorare in lista in ordine este sau nu palindrom. Functia va returna 1 daca este palindrom si 0 in caz contrar.

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