C:\Users\bilg\Documents\NetBeansProjects\CppApplication_3\main.cpp

```
#include <cstdlib>
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;


class Hata:public exception{
private:
   char* msj;
public:
   Hata(char * msj="Hatalı durum"){
      this->msj=msj;
   }
   const char * what() const throw(){
      return msj;
   }

};
class GecersizKonumHatasi:public exception{
private:
   char* msj;
public:
   GecersizKonumHatasi(char * msj= "Geçersiz konum hatası"){
      this->msj=msj;
   }
   const char * what() const throw(){
      return msj;
   }

};
class BosListeHatasi:public exception{
private:
   char* msj;
public:
   BosListeHatasi(char * msj="Boş liste hatası"){
      this->msj=msj;
   }
   const char * what() const throw(){
      return msj;
   }

};

template <typename Nesne>
class Node{
private:
   Nesne data;//veri
   Node * next;//sonraki
public:
   Node(const Nesne& data, Node<Nesne> * next=NULL){
      this->data=data;
      this->next=next;
   }
```

```
  template <typename T> friend class LinkedList;
  template <typename T> friend class ListIterator;
};


template <typename T>
class ListIterator{//liste üzerinde gezmek için
private:
   Node<T> *simdiki;
public:
   ListIterator(){
      simdiki=NULL;
   }
   ListIterator(Node<T> *simdiki){
      this->simdiki=simdiki;
   }
   void ilerle(){
      if (simdiki==NULL) throw Hata("Liste Sonu");
      simdiki=simdiki->next;
   }
   T getir(){//const T& getir(){
      return simdiki->data;
   }

   bool sonaGeldiMi(){
      return simdiki==NULL;
   }
   template <typename U> friend class LinkedList;
};

template <typename Nesne>
class LinkedList{
private:
   Node<Nesne> *head;//ilk elemanın adresi
public:
   LinkedList(){
      head=NULL;
   }


   ListIterator<Nesne> ilk()throw(BosListeHatasi){
      if (head==NULL) throw BosListeHatasi();
      return ListIterator<Nesne>(head);
   }

   void insert(int konum,const Nesne& data)throw(GecersizKonumHatasi){
      cout<<"length="<<length()<<endl;
      if (konum<0|konum>length()) throw GecersizKonumHatasi();
      //yeni düğüm oluştur
      if (konum==0){//liste başına ekle
         push_front(data);
      }else{
         Node<Nesne> *yeni=new Node<Nesne>(data);
         int sayac=0;
         Node<Nesne> *temp=head;
         while(temp->next!=NULL){
            if ( (konum-1)==sayac) break;
```

```
            temp=temp->next;
            sayac++;
        }
        yeni->next=temp->next;
        temp->next=yeni;
    }
}

void push_front(const Nesne& data){ //liste başına ekle
    Node<Nesne> *yeni=new Node<Nesne>(data);
    if (head==NULL){
        head=yeni;
    }else{
        yeni->next=head;
        head=yeni;
    }
}
void push_back(const Nesne& data){//liste sonuna ekle
    Node<Nesne> *yeni=new Node<Nesne>(data);
    if (head==NULL){
        head=yeni;
    }
    else{
        Node<Nesne> *temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=yeni;
    }
}

void remove(int konum) throw(GecersizKonumHatasi,BosListeHatasi){ //belirtilen konumdakini çıkar
    if ( konum<0|konum>(length()-1)) throw GecersizKonumHatasi();
    if (head==NULL ) throw BosListeHatasi();
    //konumu bul
    Node<Nesne> *temp;
    temp=head;
    int sayac=0;
    if (konum==0){
        head=head->next;
        delete temp;
    }
    else{
        Node<Nesne> *eskidugum;
        while (temp->next!=NULL){//konumun bir öncesi
            if (sayac==(konum-1)){
                //düğümü boşa çıkart
                eskidugum=temp->next;
                temp->next=eskidugum->next;//NULL olabilir
                //düğümü sil
                delete eskidugum;
                break;
            }
            temp=temp->next;
            sayac++;
        }
    }
```

```
}
 Nesne at(int konum)throw (Hata){//const Nesne& at(int konum)const throw (Hata){
   if (konum<0) throw Hata();   //konum geçerli değilse throw exception
   //konumu bul
   Node<Nesne> *temp;
   temp=head;
   int sayac=0;
   while (temp!=NULL){
      if (sayac==konum){
         return temp->data;
      }
      temp=temp->next;
      sayac++;
   }
   //veri bulunamadı, konum geçerli değil
   throw Hata("Geçersiz konum");
}
void yazdir(){
   cout<<"while ile yazdır"<<endl;
   Node<Nesne>* temp=head;
   while(temp!=NULL){
      cout<<temp->data<<endl;
      temp=temp->next;
   }
}
int length(){
   Node<Nesne> *temp;
   temp=head;
   int sayac=0;
   while (temp!=NULL){
      temp=temp->next;
      sayac++;
   }
   return sayac;
}
void clear(){
   //listenin elemanlarını temizle
   if (head==NULL) return;
   Node<Nesne> *temp,*cop;
   temp=head;
   int sayac=0;
   while (temp!=NULL){
      cop=temp;
      temp=temp->next;
      delete cop;
      sayac++;
   }
   head=NULL;
}

void clear2(){
   //listenin elemanlarını temizle
   if (head==NULL) return;
   Node<Nesne> *temp;
   int sayac=0;
   while (head!=NULL){
      temp=head;
```

```
            head=head->next;
            delete temp;
            sayac++;
        }
    }

    bool empty(){
        return head==NULL;
    }

};

template <typename T>
void yazdir(LinkedList<T> *liste1){
    cout<<"--liste----------"<<endl;
    for (int i = 0; i < liste1->length(); i++) {
        cout<<liste1->at(i)<<endl;
    }
}

int main(int argc, char** argv)
{
    //LinkedList<int> *liste1=new LinkedList<int>;
    LinkedList<string> *liste2=new LinkedList<string>;
    try{

        liste2->insert(0,"deneme1");
        liste2->insert(0,"deneme2");
        liste2->insert(0,"deneme3");
        liste2->insert(0,"deneme4");

        //liste2->at(2)="degistir";

        //ListIterator<string> itr=liste2->ilk();

        for(ListIterator<string> itr=liste2->ilk();!itr.sonaGeldiMi();itr.ilerle()){

            cout<<itr.getir()<<endl;

        }
        liste2->clear();
        yazdir(liste2);

    }
    catch(exception &e){
        cout<<"Hata:"<<e.what()<<endl;
    }


    return 0;
}
```