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

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


/*
 Bu versiyonda basitleştirme için template yok
 */
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;
   }

};

class Node{
private:
   int data;//veri
   Node * next;//sonraki
public:
   Node(int data,Node * next=NULL){
      this->data=data;
```

```
        this->next=next;
    }
    friend class LinkedList;
};
class LinkedList{
private:
    Node *head;//ilk elemanın adresi
public:
    LinkedList(){
        head=NULL;
    }
    void insert(int konum,int data)throw(GecersizKonumHatasi){

        if (konum<0|konum>length()) throw GecersizKonumHatasi();
        //yeni düğüm oluştur
        if (konum==0){//liste başına ekle
            push_front(data);
        }else{
            Node *yeni=new Node(data);
            int sayac=0;
            Node *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(int data){ //liste başına ekle
        Node *yeni=new Node(data);
        if (head==NULL){
            head=yeni;
        }else{
            yeni->next=head;
            head=yeni;
        }
    }
    void push_back(int data){//liste sonuna ekle
        Node *yeni=new Node(data);
        if (head==NULL){
            head=yeni;
        }
        else{
            Node *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 *temp;
    temp=head;
    int sayac=0;
    if (konum==0){
        head=head->next;
        delete temp;
    }
    else{
        Node *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++;
        }
    }
}
int at(int konum)throw(GecersizKonumHatasi,BosListeHatasi){ //belirtilen konumdakini çıkar
    if ( konum<0|konum>(length()-1)) throw GecersizKonumHatasi();
    if (head==NULL ) throw BosListeHatasi();
    //konumu bul
    Node *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* temp=head;
    while(temp!=NULL){
        cout<<temp->data<<endl;
        temp=temp->next;
    }
}
int length(){
    Node *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 *temp,*cop;
        temp=head;
        int sayac=0;
        while (temp!=NULL){
            cop=temp;
            temp=temp->next;
            delete cop;
            sayac++;
        }
        head=NULL;
    }
    bool empty(){
        return head==NULL;
    }

};

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

int main(int argc, char** argv)
{
    LinkedList *liste1=new LinkedList;

    try{
        liste1->push_back(10);
        liste1->push_back(20);
        liste1->push_back(30);
        liste1->push_back(40);
        liste1->push_back(50);
        liste1->insert(0,111);

        yazdir(liste1);

        liste1->remove(5);

        yazdir(liste1);

       // liste1->clear();

        //yazdir(liste1);

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


    //liste1->yazdir();
```

```
    //liste1->push_back(111);
    //liste1->yazdir();

    return 0;
}
```