```
// Ex. 8.1
#include <iostream>
using namespace std;
//#include <string.h>
class CAT
{
public:
      CAT() {pNext=0;}
      ~CAT(){;}
      char * GetName() {return CatName;}
      void SetName(char name[20]) {strncpy(CatName, name,19);}
      CAT * GetNext() {return pNext;}
      void SetNext(CAT *pN) {pNext = pN;}
private:
      char CatName[20];  //CatName is a string
      CAT *pNext;
};

CAT * create(int); //Function prototype
void show (CAT *); //Function prototype
CAT * insert(int, char *, CAT *);  //Function prototype
CAT * del(int, CAT *); //Function prototype
void menu(CAT *); //Function prototype
int no_of_cat; //Global storing the current no. of cats in the list
void del_list(CAT *); //Delete the whole linked list before leaving

int main()
{
      CAT *pHead;
      cout << "How many Cats: ";
      cin >> no_of_cat;
      if (no_of_cat>0)
      {
            pHead = create(no_of_cat);
            menu(pHead);
      }
      else
            cout<<"The number of Cats must be bigger than 0; BYE!"<<endl;
      return 0;
} //end main

CAT * create(int n)  // n>0 ensured by main()
{
      char cname[20];
      int i;
      CAT *pH=NULL;
      for (i=0; i<n; i++)
      {
            cout << "Please insert the name of cat " << i <<":  ";
            cin >> cname;
            pH=insert(i, cname, pH);
      }
      return pH;
}

void show (CAT *pH_show)
{
      while (pH_show!=NULL)
      {
            cout << "CatName: " << pH_show->GetName() << endl;
            pH_show = pH_show->GetNext();
```

```
      }
}

CAT * insert(int n, char *pName, CAT *pHead)
// int n - the insert position in the linked list
// The first item is in position 0, i.e. 0-th cat
// char *pName - the name of the cat to be inserted
// CAT *pHead - the head pointer (initially pHead = NULL)
// The function should return the revised head pointer
{
      CAT *pCurr, *pPrev, *NewCat;
      int i;
      NewCat = new CAT;
      //check if the new cat is inserted as the first element.
      if (n==0)
      {
            NewCat->SetName(pName);
            NewCat->SetNext(pHead);
            return NewCat;  //Do NOT need else because of return
      }
      //search the insert point
      pCurr = pHead;
      for (i=1; i<n; i++) //pCurr points to (n-1)-th cat after for loop.
            pCurr = pCurr->GetNext();
      //It does NOT matter whether the (n-1)-th cat is the last cat.
      //add an element at n-th position
      pPrev=pCurr;
      pCurr = pCurr->GetNext();
      pPrev->SetNext(NewCat);
      NewCat->SetName(pName);
      NewCat->SetNext(pCurr);
      return pHead;
}

CAT * del(int n, CAT *pHead)
{
// int n - the delete position in the linked list
// CAT *pHead - the head pointer
// The function should return the revised head pointer
      CAT *pCurr, *pPrev, *pCat;
      int i;
//Check if the first element is to be deleted.
      if (n==0)
      {
            pCurr = pHead->GetNext();
            delete pHead;
            return pCurr;  //Do NOT need else because of return
      }
//delete another element except the first one
      pCurr = pHead;
      for (i=0; i<n; i++) //pPrev points to (n-1)-th cat after for loop.
      {
            pPrev = pCurr;
            pCurr = pPrev->GetNext();
      }  //pCurr points to n-th cat
      //delete an element at n position
      pCat=pCurr->GetNext();
      pPrev->SetNext(pCat);
      delete pCurr;
      return pHead;
}
```

```
void menu(CAT * Head)
{
	int option, n;
	do
	{ //display menu
		cout << "[1] Show List" << endl;
		cout << "[2] Insert a cat" << endl;
		cout << "[3] Delete a cat" << endl;
		cout << "[4] Quit" << endl;
		//get selected option
		cin >> option;
		switch(option){
		case 1: //Show cat list
			show(Head);
			break;
		case 2: //insert element
			cout << "Position of the new cat to be inserted: ";
			cin >> n;
			if (n>=0 && n<=no_of_cat) //n=no_of_cat means add to end
			{
				char newname[20];
				cout << "Name of the new cat ";
				cin >> newname;
				Head=insert(n, newname, Head);
				no_of_cat++;
			}
			else cout<<"Position outside the linked list!\n";
			break;
		case 3:
			cout << "Position of the cat to be deleted: ";
			cin >> n;
			if (n>=0 && n<no_of_cat)
			{
				Head = del(n, Head); //Avoid keyword delete
				no_of_cat--;
			}
			else cout<<"Position outside the linked list!\n";
			break;
		case 4:
			break;
		default:
			cout<< "Please enter [1] to [4]!\n"; break;
		} //End case
	}while(option !=4);  //End do while
}

void del_list(CAT *pH)  // Delete the whole linked list before leaving
{
	CAT *pT, *pCat;
	pT=pH;
	do
	{
		pCat=pT->GetNext();
		delete pT;
		pT=pCat;
	}
	while (pT!=NULL);
}
```