main.cpp

```cpp
#include "df.h"



int main()
{
    SqList L;
    ElemType stu,stu_tmp;
    int cases;
    ll index;
    if(!InitList(L))
    {
        cout << "OVERFLOW!" << endl;
        exit(-1);
    }
    while (true)
    {
        BeginText();
        cin >> cases;
        switch (cases)
        {
            case 0:
                exit(-1);
                break;
            case 1:
                cout<<"Input number name age department."<<endl;
                CreateElem(stu);
                isDoErrorText(ListInsert(L, ListLength(L) + 1, stu));
                break;
            case 2:
                cout<<"Delete student information according to numbers."<<endl;
                cin>>stu.StudentNumber;
                index = LocateElem(L, stu, EQUAL);
                isDoErrorText(index && ListDelete(L, index, stu));
                break;
            case 3:
                cout<<"Renew student name according to numbers."<<endl;
                cin>>stu.StudentNumber;
                index = LocateElem(L, stu, EQUAL);
                if (index > 0)
                {
                    cout<<"Input new name:"<<endl;
                    cin>>stu.StudentName;
                    GetElem(L,index,stu_tmp);
                    stu.StudentAge = stu_tmp.StudentAge;
                    strcpy_s(stu_tmp.StudentName,stu.StudentName);
                    isDoErrorText(PutElem(L,index,stu));
                }
                else
```

```
                    isDoErrorText(NOTFOUND);
                break;
            case 4:
                cout<<"Input number to find the student."<<endl;
                cin>>stu.StudentNumber;
                index = LocateElem(L, stu, EQUAL);
                ElemType tempe;
                if (index)
                    PrintElem((GetElem(L,index,tempe),tempe));
                else
                    isDoErrorText(NOTFOUND);
                break;
            case 5:
                isDoErrorText(ListTraverse(L,VISIT));
                break;
            case 6:
                cout<<ListLength(L)<<endl;
                break;
        }
    }
    return 0;
}
```

df.h

```
#include "iostream"
#include <cstdio>
#include <cstring>
#include <cstdlib>

using namespace std;

#define ll long long
#define scanf scanf_s
#define MAXSIZE 20

#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE (-1)
#define OVER (-2)

#define NOTFOUND 2

#define FINDALL 0
#define FINDNUM 1
#define EQUAL 1
#define WRONGMODEL 3
#define VISIT 0
```

```
typedef int Status;

typedef struct {
    char StudentNumber[20];
    char StudentName[20];
    int StudentAge;
    char StudentDepartment[10];
}ElemType;

typedef struct Node   //链表的结点
{
    ElemType data;
    struct Node *next;
}Node,*SqList;



bool CharCompare(char a[],char b[]);
Status InitList(SqList &L);
Status DestoryList(SqList &L);
Status ClearList(SqList &L);
bool ListEmpty(SqList L);
ll ListLength(SqList L);
Status GetElem(SqList L, ll i, ElemType &e);
Status PutElem(SqList &L,ll i,ElemType &e);
bool compare(ElemType a, ElemType b);
ll LocateElem(SqList L, ElemType e,Status s);
Status ListInsert(SqList &L, ll i, ElemType e);
Status ListDelete(SqList &L, ll i, ElemType &e);
Status ListTraverse(SqList L,Status t);
void PrintElem(ElemType e);
void CreateElem(ElemType &e);
ll FindNumber(SqList L,ElemType e);
void BeginText();
void isDoErrorText(Status t);
```

ADT.cpp

```
#include"df.h"

using namespace std;

bool CharCompare(char a[],char b[])
{
    ll len_a = strlen(a);
    ll len_b = strlen(b);
    if (len_a != len_b)
        return false;
    for (ll i = 0;i<len_b;i++)
        if (a[i] != b[i])
```

```
            return false;
    return true;
}

Status InitList(SqList &L)
{
    L = new Node;
    L->next = NULL;
    return OK;
}

Status DestoryList(SqList &L)
{
    L->next = NULL;
    return OK;
}

Status ClearList(SqList &L)
{
    if ( !ListLength(L) )
    {
        InitList(L);
        return OK;
    }
    return ERROR;
}

bool ListEmpty(SqList L)
{
    return bool(ListLength(L));
}

ll ListLength(SqList L)
{
    ll len=0;
    SqList temp = L;
    while (temp)
    {
        len++;
        temp=temp->next;
    }
    return len - 1;
}

Status GetElem(SqList L, ll i, ElemType &e)
{
    if ( i!=0 && ListLength(L)<i)
        return ERROR;
    SqList temp = L;
```

```
    for (int t = 0;t<i;t++)
        temp = temp->next;
    e = temp->data;
    return OK;
}

bool compare(ElemType a, ElemType b)
{
    return (a.StudentAge == b.StudentAge) || CharCompare(a.StudentNumber,b.StudentNumber)
|| CharCompare(a.StudentDepartment,b.StudentDepartment) ||
           CharCompare(a.StudentName,b.StudentName);
}

ll LocateElem(SqList L, ElemType e,Status s)
{
    if (s == FINDALL)
    {
        for (ll i = 0; i < ListLength(L); i++)
            if (compare(L->data, e))
                return i + 1;
        return NOTFOUND;
    }
    if (s == EQUAL)
        return FindNumber(L,e);
    return WRONGMODEL;
}

Status ListInsert(SqList &L, ll i, ElemType e)
{
    SqList temp = L;
    for(int t = 1;t< i; t++)
        temp = temp->next;
    SqList s = new Node;
    s->data = e;
    if (i != ListLength(L))
        s->next = temp->next;
    temp->next = s;
    return OK;
}

Status ListDelete(SqList &L, ll i, ElemType &e)
{
    SqList temp = L;
    for(int t = 1;t<i;t++)
        temp = temp->next;
    Node *p = temp->next;
    if (i != 0)
        temp->next = p->next;
    else
```

```
        p=temp->next;
    delete p;
    return OK;
}
Status ListTraverse(SqList L,Status t)
{
    switch (t)
    {
        case VISIT:
            SqList temp = L;
            if (ListLength(temp))
            {
                temp = temp->next;
                for (ll i = 0; i < ListLength(L); i++)
                {
                    PrintElem(temp->data);
                    temp = temp->next;
                }
            }
            else
                return ERROR;
            return OK;
            break;
    }
    return ERROR;
}

Status PutElem(SqList &L,ll i,ElemType &e)
{
    if (i == 0 || i > ListLength(L))
        return ERROR;
    SqList temp = L;
    for (int t = 0;t<i;t++)
        temp = temp->next;
    temp->data = e;
    return OK;
}



void PrintElem(ElemType e)
{
    cout<<"Number \t\t"<<e.StudentNumber<<endl;
    cout<<"Name \t\t"<<e.StudentName<<endl;
    cout<<"Age \t\t"<<e.StudentAge<<endl;
    cout<<"Department \t"<<e.StudentDepartment<<endl<<endl;
}

void CreateElem(ElemType &e)
{
```

```
    cin>>e.StudentNumber>>e.StudentName>>e.StudentAge>>e.StudentDepartment;
}

ll FindNumber(SqList L,ElemType e)
{
    ll pos = 0;
    int flag = 0;
    SqList temp = L;
    temp = temp->next;
    for (ll p = 0;p< ListLength(L);p++)
    {
        pos = pos + 1;
        if (CharCompare(temp->data.StudentNumber, e.StudentNumber))
        {
            flag = 1;
            break;
        }
        temp = temp->next;
    }
    return flag?pos:0;
}
void BeginText()
{
    cout<<"Please select the function:"<<endl;
    cout<<"0\texit"<<endl;
    cout<<"1\tinsert student"<<endl;
    cout<<"2\tdelete student"<<endl;
    cout<<"3\tupdate student name"<<endl;
    cout<<"4\tsearch student with number"<<endl;
    cout<<"5\tshow all student"<<endl;
    cout<<"6\tcount the student"<<endl;
    cout<<"Please input:"<<endl;
}

void isDoErrorText(Status t)
{
    switch (t)
    {
        case OK:
            cout<<"<SUCCESS>"<<endl;
            break;
        case INFEASIBLE:
            cout<<"<UNSUCCESS>Fine the same number in SqList."<<endl;
            break;
        case ERROR:
            cout<<"<UNSUCCESS>Input number is longer than SqList range."<<endl;
            break;
        case OVER:
            cout<<"<UNSUCCESS>SqList has already been full."<<endl;
```

```
            break;
        case NOTFOUND:
            cout<<"<UNSUCCESS>Student can't be found."<<endl;
            break;
        case WRONGMODEL:
            cout<<"<UNSUCCESS>A wrong model is given."<<endl;
            break;
    }
}
```