Tongji University

Data Structure Course Curriculum Design Document

Task1: Exam Registration System

Author:
嘉杰 李
Jiajie Li

Supervisor:
颖 张
Prof. Ying Zhang

A companion use documentation for a program submitted in fulfillment of the requirements for the Data Structure curriculum design

in the

School of Software Engineering, Tongji

December 27, 2019

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

Contents

Chapter 1

Analysis

1.1 Overview

With the development of society, with the joint efforts of all the teachers and students of the college, the scale of the school has been continuously expanded, and relevant information has been extracted from daily teaching activities to reflect the teaching situation. The traditional manual operation method is prone to data loss, statistical errors, high labor intensity, and slow speed. Using a computer can complete the above tasks quickly and quickly. After the computer is networked, the data is transmitted on the Internet, which can realize data sharing, avoid repeated labor, and standardize teaching management behaviors, thereby improving management efficiency and level. The student status management information system uses a computer as a tool, and through the information management required for educational administration management, it frees administrators from tedious data calculation and processing, so that they have more energy to engage in the research and implementation of educational administration management policies.

1.2 Background Analysis

The exam registration system is an indispensable part of a school. It is vital for school administrators and students, so a good exam registration system should be able to provide users with sufficient information and functions. The examination registration system is extremely important for schools to strengthen examination management. With the increasing number of students and the number of exams, how to manage such a huge amount of data seems extremely complicated. Traditional manual management workload is large and error-prone.

With the continuous maturity of computer science and technology, the use of computers to manage the examination registration system has advantages that manual management cannot match. These advantages can greatly improve the efficiency of schools and students, and are also important conditions for schools to move towards information, science and internationalization. Therefore, it is of great significance to develop a test registration system.

1.3 Functional Analysis

As a simplest test registration system, the first function that should be available is to enter the students' test registration status and display it. Second, the test registration system should also have insert, delete, and modify functions to ensure that students can change their test registration at any time. Finally, the test registration system software should also ensure that the software can be shut down properly.

In summary, an examination registration system should have at least the functions of input, output, insert, delete, modify, and exit.

Chapter 2

Design

2.1 Data structure design

As described in the functional analysis above, the system requires a large number of insert, delete, and modify operations, and the linked list is very simple to perform operations such as inserting and deleting, so consider using a linked list data structure.

At the same time, in order to achieve simplicity, a head node is added before the first node, so that adding or deleting head nodes is the same as processing other nodes, and the program is concise.

2.2 Class structure design

I implemented the template linked list myself according to the SGI_STL linked list standard and placed it in ljj_linkedlist.h, including two abstract data types-linked list node class (___listNode) and linked list class (LinkList), and the coupling relationship between the two classes Multiple relationships such as nesting and inheritance can be used. In order to facilitate processing, the system uses a template struct to describe the linked list node class (___listNode), so that the template linked list node class (LinkList) can access the linked list node.

2.3 Member function design

Datanode for student information

```
class datanode{
   public:
      bool avail=true;
      int id, age;
      string s_id, s_age; //should be integer
      string name, cat, gender;
   };
```

ListNode Class:

```
template <class T> struct __listNode {
   __listNode<T>* prev;
   __listNode<T>* next;
   T data;
};
```

Linked List Class:

```
template <class T> class list{
   private:
      typedef __listNode<T> list_node;
      typedef list_node* link_type;
      link_type get_node(){
         link_type p=(link_type)malloc(sizeof(list_node));
         p->next=NULL;
         p->prev=NULL;
         return p;
      }
      void put_node(link_type p){
         p->prev=NULL;
         p->next=NULL;
         p=NULL;
         free(p);
      }
      link_type create_node(const T& x){
         link_type p = get_node();
         p->data = x;
         return p;
      }
      void destroy_node(link_type p){
         put_node(p);
      }
      void __insert(link_type p, const T &x){
         len++;
         link_type temp = create_node(x);
         p->next->prev = temp;
         temp->next = p->next;
         temp->prev = p;
         p->next = temp;
      }
      void __del(link_type p){
         len--;
         p->prev->next=p->next;
         p->next->prev=p->prev;
         destroy_node(p);
      }
      link_type loc(int pos){
         pos+=1;
```

```
        if (pos>len) return NULL;
        link_type temp = hnode;
        while (pos--) temp=temp->next;
        return temp;
    }
    link_type loc_by_id(int id){
        auto temp = hnode->next;
        while (temp!=hnode) {
            if (temp->data.id==id) return temp;
            temp=temp->next;
        }
        return NULL;
    }

public:
    list();
    ~list();
    int Class_test(T temp);
    int push_back(const T& x) {__insert(hnode->prev, x); return 0;}
    int empty();
    int plist();
    bool isEmpty() {return hnode==hnode->next;}

    int del(int pos);
    int find_by_id(int id);
    int del_by_id(int id); // delete a node in the linked list
    int insert(int pos, const T &x);
    int modify(int pos, const T &x);
    int modify_by_id(int id, const T &x);

    int len;
    link_type hnode; //head node
};
```

2.4 System design

The system first calls the inputdata function to complete the creation of the linked list datasystem and input data, and then executes the member functions corresponding to the linked list datasystem according to the operation code (code) entered by the user.

Chapter 3

Implementation

3.1 Insertion function implementation

3.1.1 Insert function core code

```
void __insert(link_type p, const T &x){
    len++;
    link_type temp = create_node(x);
    p->next->prev = temp;
    temp->next = p->next;
    temp->prev = p;
    p->next = temp;
}
```

3.1.2 Insert function flowchart

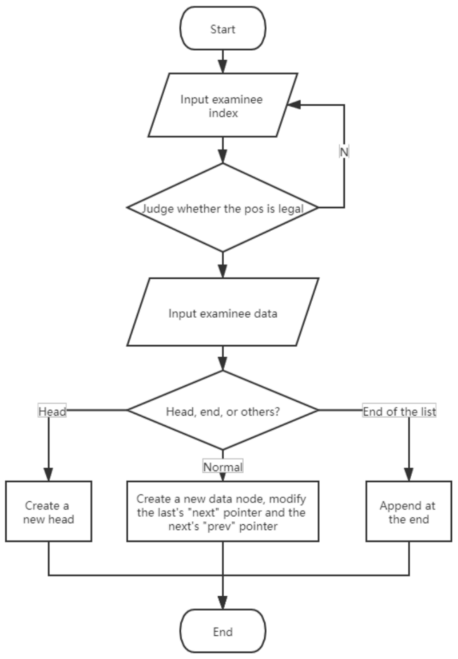
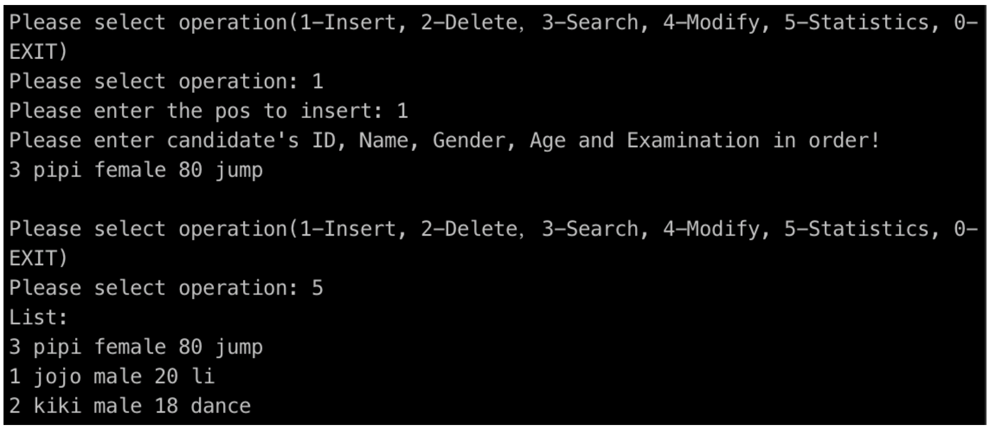


Figure 3.1: Insert function flowchart

3.1.3 Insert function screenshot example



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 1
Please enter the pos to insert: 1
Please enter candidate's ID, Name, Gender, Age and Examination in order!
3 pipi female 80 jump

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
3 pipi female 80 jump
1 jojo male 20 li
2 kiki male 18 dance
```

Figure 3.2: Insert function screenshot example

3.2 Deletion function implementation

3.2.1 Delete function core code

```
void __del(link_type p){
    len--;
    p->prev->next=p->next;
    p->next->prev=p->prev;
    destroy_node(p);
}
```

3.2.2 Delete function flowchart

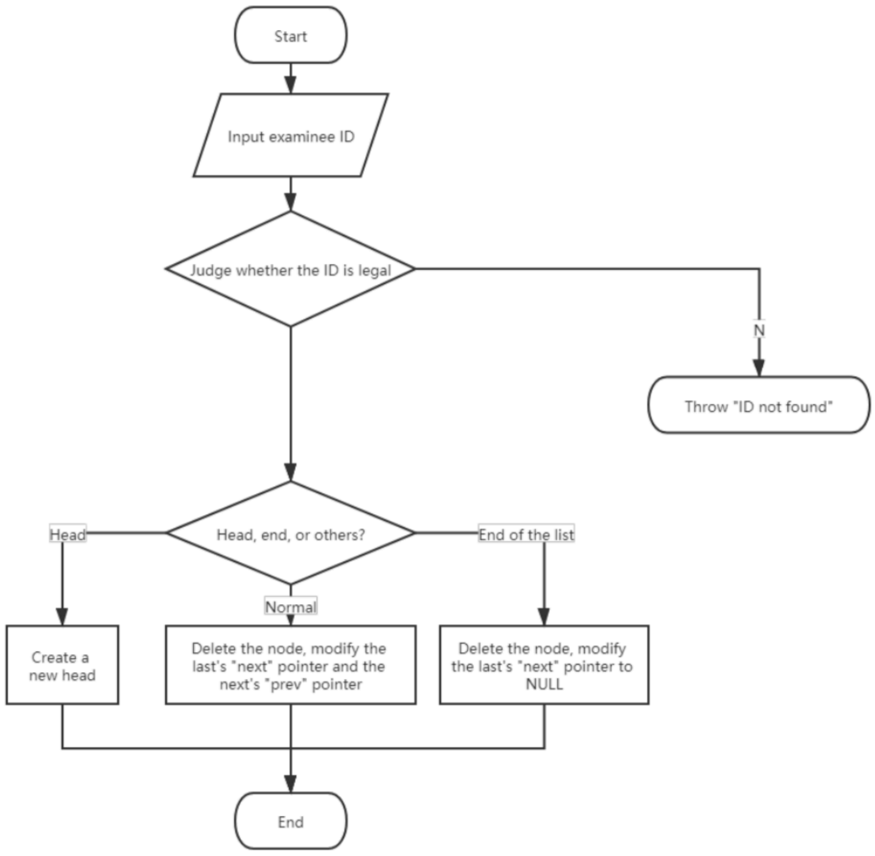
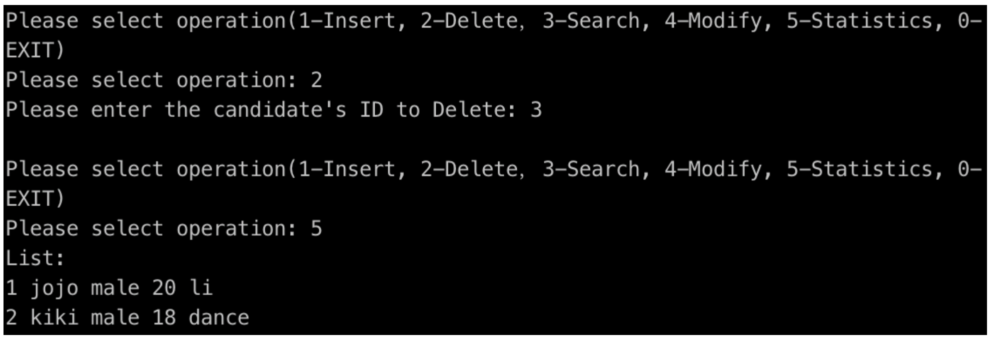

Start
Input examinee ID
Judge whether the ID is legal
N
Throw "ID not found"
Head, end, or others?
Head
End of the list
Normal
Create a new head
Delete the node, modify the last's "next" pointer and the next's "prev" pointer
Delete the node, modify the last's "next" pointer to NULL
End

Figure 3.3: Delete function flowchart

3.2.3 Delete function screenshot example



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 2
Please enter the candidate's ID to Delete: 3

Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance
```

Figure 3.4: Delete function screenshot example

3.3 Find function implementation

3.3.1 Find function core code

```
link_type loc_by_id(int id){
    auto temp = hnode->next;
    while (temp!=hnode) {
        if (temp->data.id==id) return temp;
        temp=temp->next;
    }
    return NULL;
}
```

3.3.2 Find function flowchart

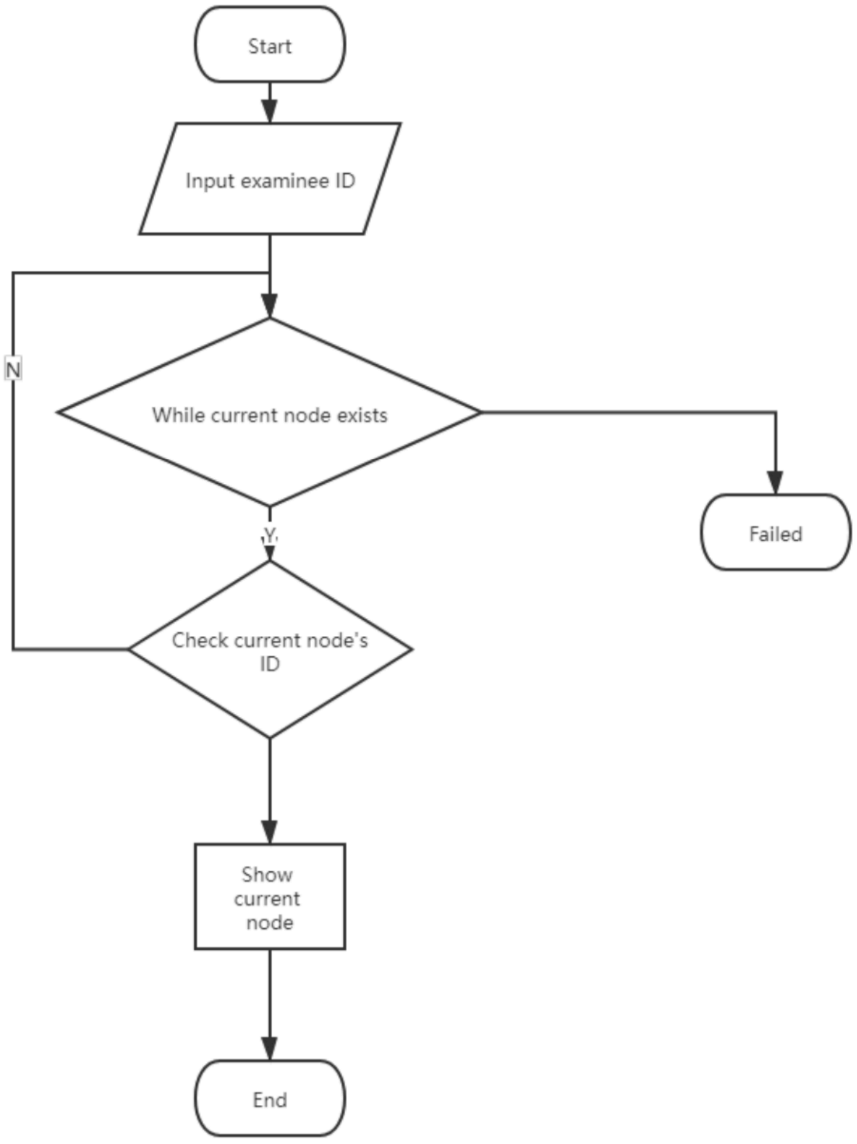
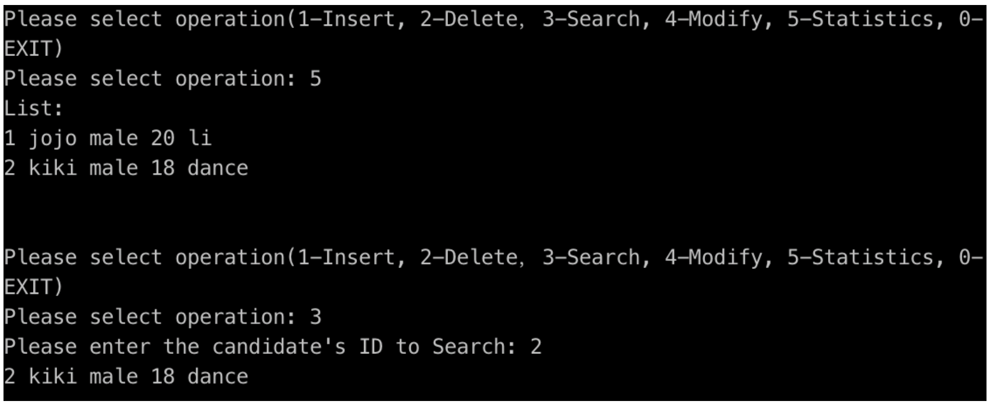


Figure 3.5: Find function flowchart

3.3.3 Find function screenshot example



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance


Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 3
Please enter the candidate's ID to Search: 2
2 kiki male 18 dance
```

Figure 3.6: Find function screenshot example

3.4 Modification function implementation

3.4.1 Modify function core code

```
    int list <T>::modify_by_id(int id, const T &x) {
    link_type p = loc_by_id(id);
    if (p==NULL) {
        cerr << "Error: ID not Found!" << std::endl;
        return 1;
    }
    p->data=x;
    return 0;
}
```

3.4.2 Modify function flowchart

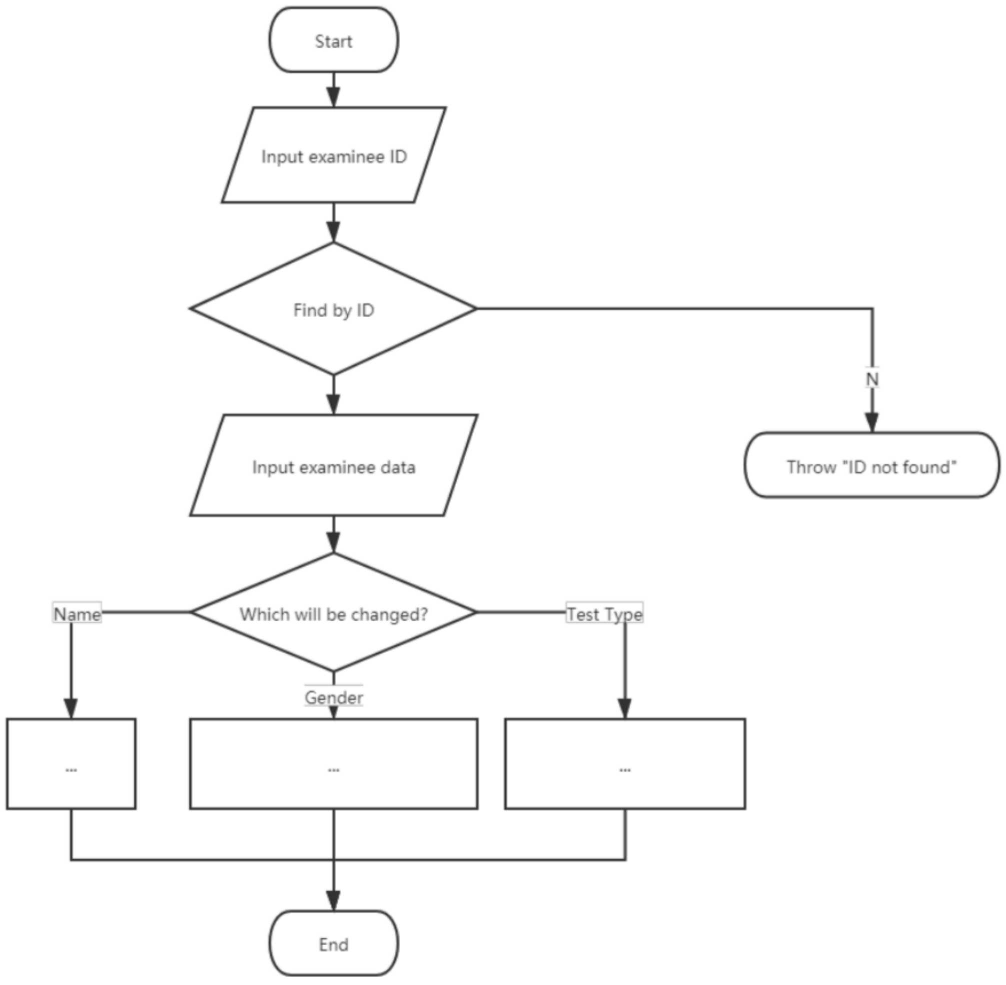
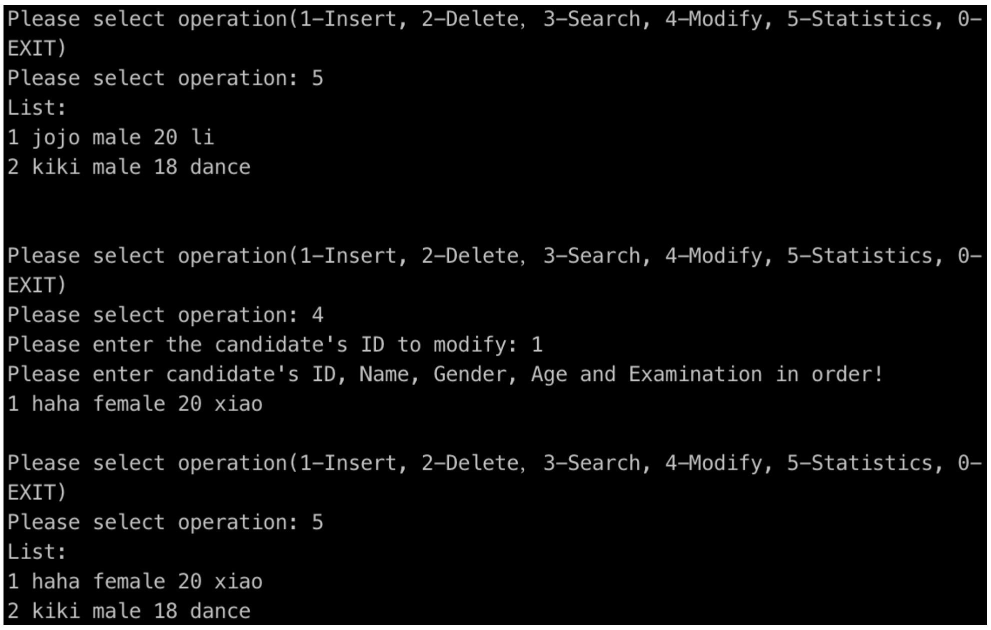


Figure 3.7: Modify function flowchart

3.4.3 Modify function screenshot example



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance


Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 4
Please enter the candidate's ID to modify: 1
Please enter candidate's ID, Name, Gender, Age and Examination in order!
1 haha female 20 xiao

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
1 haha female 20 xiao
2 kiki male 18 dance
```

Figure 3.8: Modify function screenshot example

3.5 Listing function implementation

3.5.1 List function core code

```
    int list <T>::plist(){
    auto temp = hnode->next;
    std::cout << "List:" << std::endl;
    while (temp!=hnode) {
        //std::cout << temp->data.age << " ";
        temp->data.pdata();
        temp = temp->next;
    }
    std::cout << std::endl;
    return 0;
}
```

3.5.2 List function flowchart

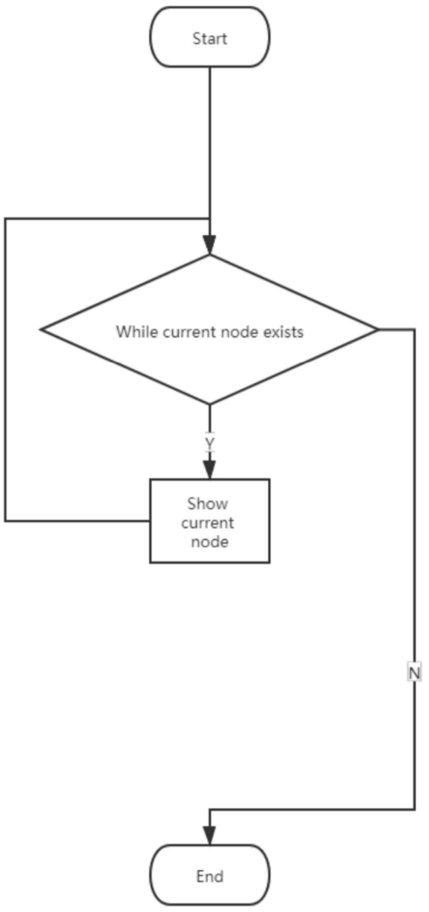
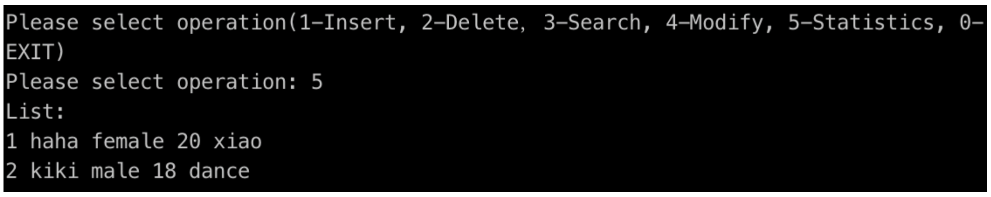


Figure 3.9: List function flowchart

3.5.3 List function screenshot example



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 haha female 20 xiao
2 kiki male 18 dance
```

Figure 3.10: List function screenshot example

Chapter 4

Test

4.1 Function Tests

4.1.1 Establish

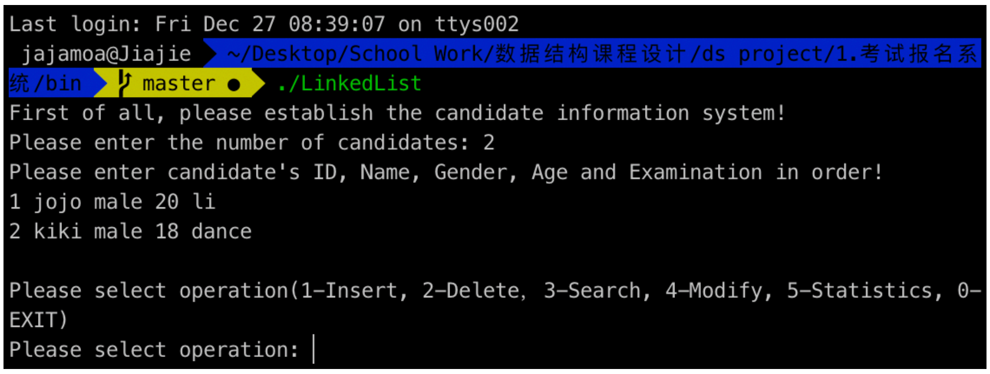
Test Case:

establish the candidate information system

Expected Result:

print ”Please select operation...”

Program Result:



```
Last login: Fri Dec 27 08:39:07 on ttys002
 jajamoa@Jiajie  ~/Desktop/School Work/数据结构课程设计/ds project/1.考试报名系
统/bin  ᚴ master ●  ./LinkedList
First of all, please establish the candidate information system!
Please enter the number of candidates: 2
Please enter candidate's ID, Name, Gender, Age and Examination in order!
1 jojo male 20 li
2 kiki male 18 dance

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: |
```

Figure 4.1: Screenshots

4.1.2 Insertion

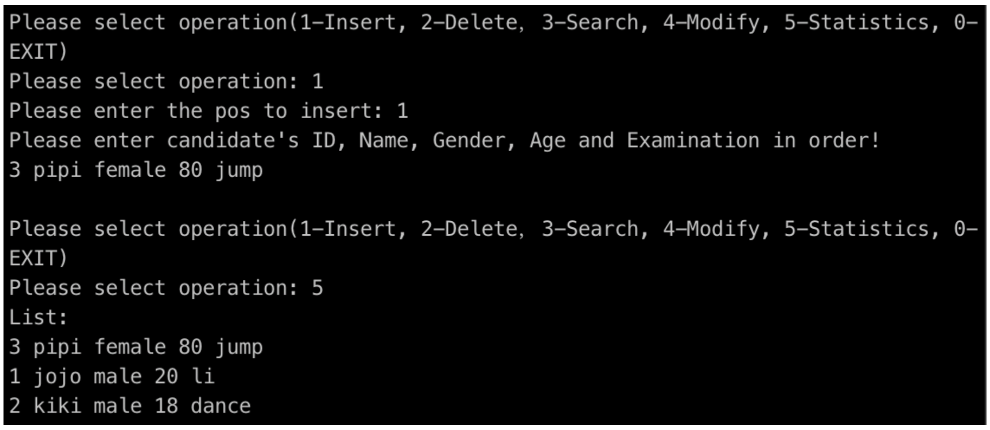
Test Case:

insert new datanode to information system

Expected Result:

new datanode added to expected position

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 1
Please enter the pos to insert: 1
Please enter candidate's ID, Name, Gender, Age and Examination in order!
3 pipi female 80 jump

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
3 pipi female 80 jump
1 jojo male 20 li
2 kiki male 18 dance
```

Figure 4.2: Screenshots

4.1.3 Deletion

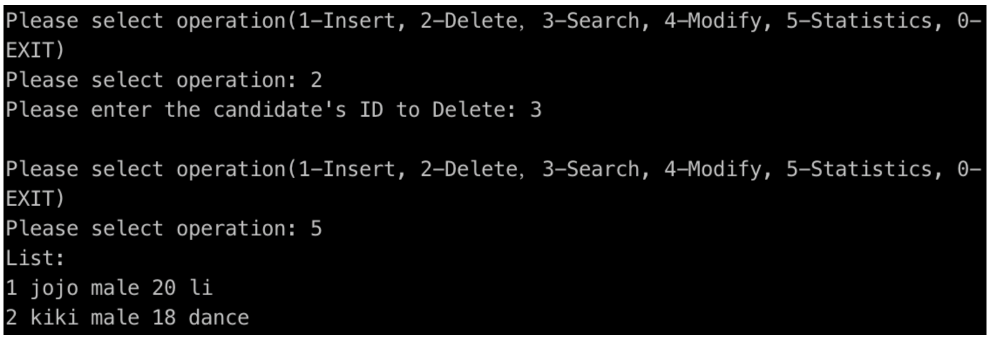
Test Case:

delete one node of the candidate information system

Expected Result:

seleceted node deleted

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 2
Please enter the candidate's ID to Delete: 3

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance
```

Figure 4.3: Screenshots

4.1.4 Find

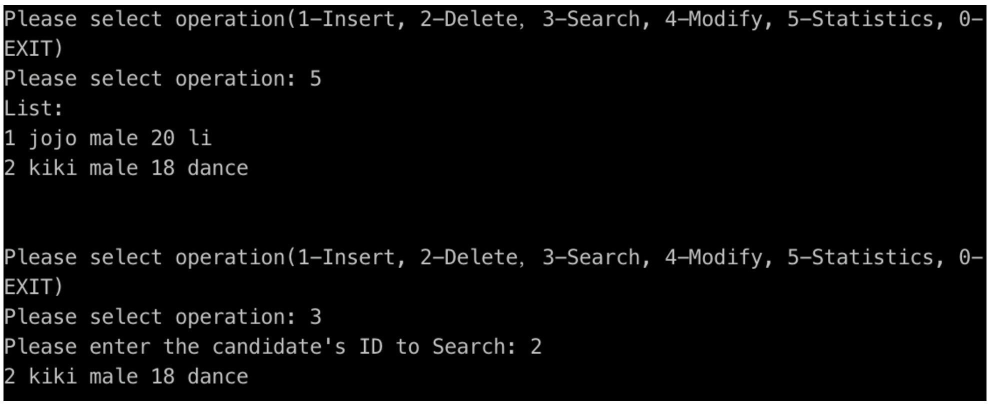
Test Case:

find a node by id in the candidate information system

Expected Result:

print searched node info

Program Result:



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance


Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 3
Please enter the candidate's ID to Search: 2
2 kiki male 18 dance
```

Figure 4.4: Screenshots

4.1.5 Modification

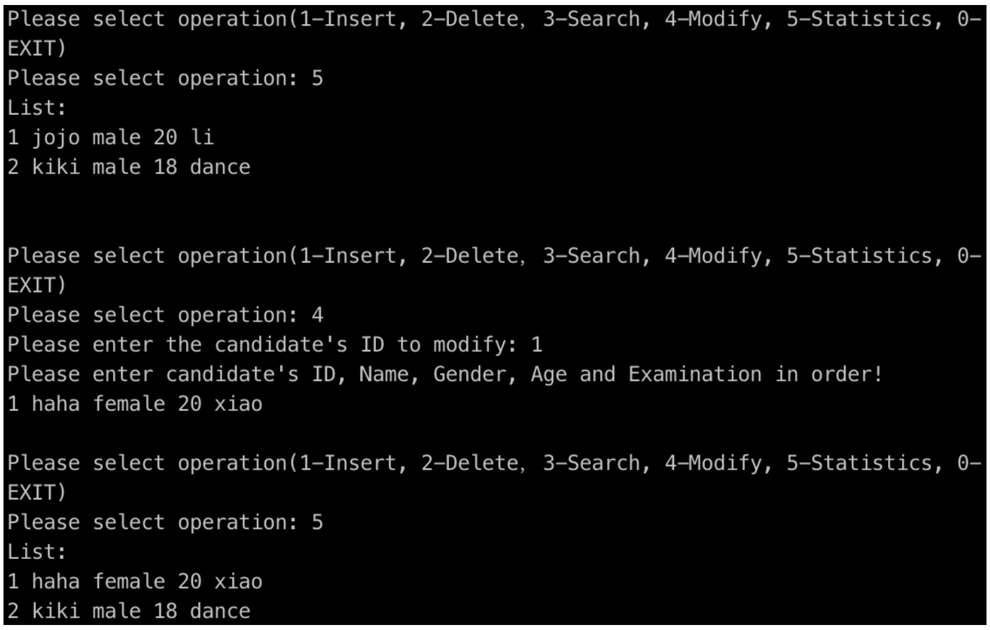
Test Case:

modify information of a node in the candidate information system

Expected Result:

selecteed node's info modified

Program Result:



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 jojo male 20 li
2 kiki male 18 dance


Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 4
Please enter the candidate's ID to modify: 1
Please enter candidate's ID, Name, Gender, Age and Examination in order!
1 haha female 20 xiao

Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 haha female 20 xiao
2 kiki male 18 dance
```

Figure 4.5: Screenshots

4.1.6 Listing

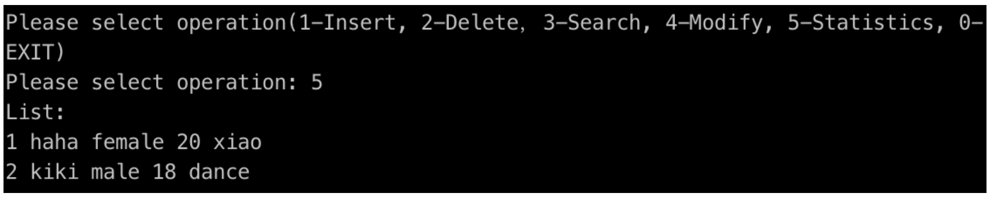
Test Case:

print the candidate information system

Expected Result:

print the candidate information system

Program Result:



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 5
List:
1 haha female 20 xiao
2 kiki male 18 dance
```

Figure 4.6: Screenshots

4.2 Boundary Tests

4.2.1 No input data

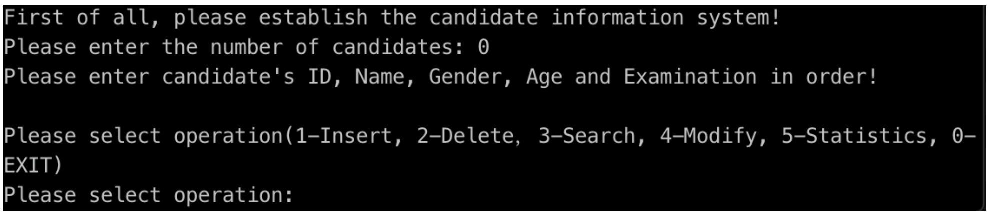
Test Case:

no input data

Expected Result:

The program runs normally without crashing

Program Result:



```
First of all, please establish the candidate information system!
Please enter the number of candidates: 0
Please enter candidate's ID, Name, Gender, Age and Examination in order!

Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation:
```

Figure 4.7: Screenshots

4.2.2 Delete head node

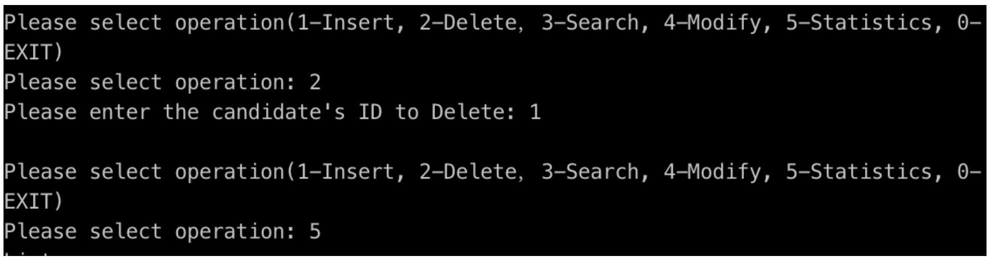
Test Case:

delete head node

Expected Result:

The program runs normally without crashing

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 2
Please enter the candidate's ID to Delete: 1

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
```

Figure 4.8: Screenshots

4.2.3 Empty list after deleting

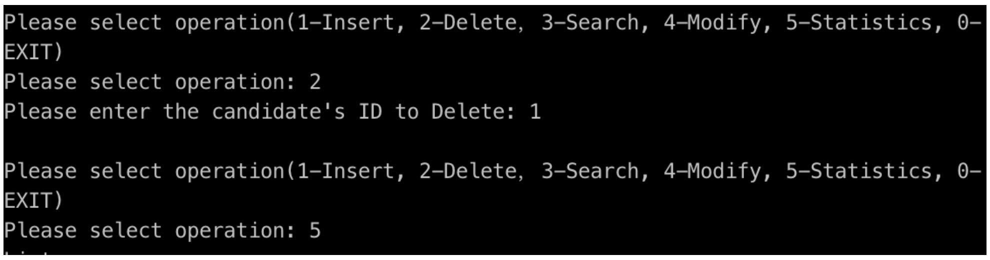
Test Case:

empty list after deleting

Expected Result:

The program runs normally without crashing

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 2
Please enter the candidate's ID to Delete: 1

Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 5
```

Figure 4.9: Screenshots

4.3 Exception

4.3.1 Input invlaid information

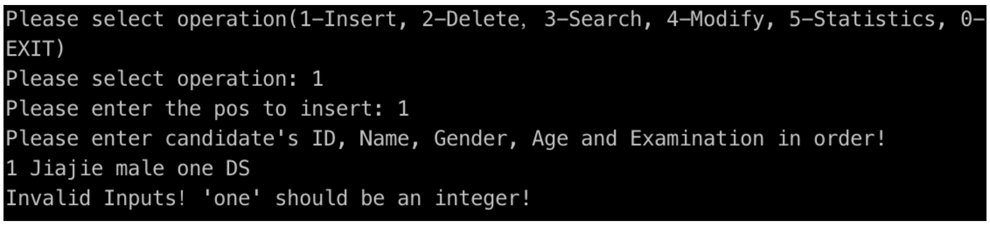
Test Case:

input an ”one” as age

Expected Result:

print ”Invalid Inputs! ’one’ should be an integer!”

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 1
Please enter the pos to insert: 1
Please enter candidate's ID, Name, Gender, Age and Examination in order!
1 Jiajie male one DS
Invalid Inputs! 'one' should be an integer!
```

Figure 4.10: Screenshots

4.3.2 Opcode error

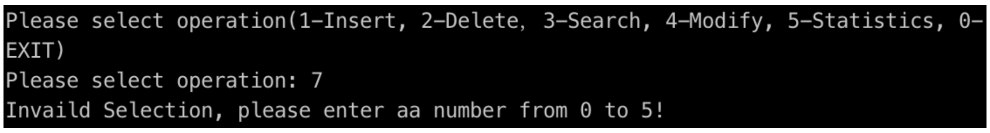
Test Case:

input an ”7” as operation

Expected Result:

print ”Invaild Selection, please enter aa number from 0 to 5!”

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 7
Invaild Selection, please enter aa number from 0 to 5!
```

Figure 4.11: Screenshots

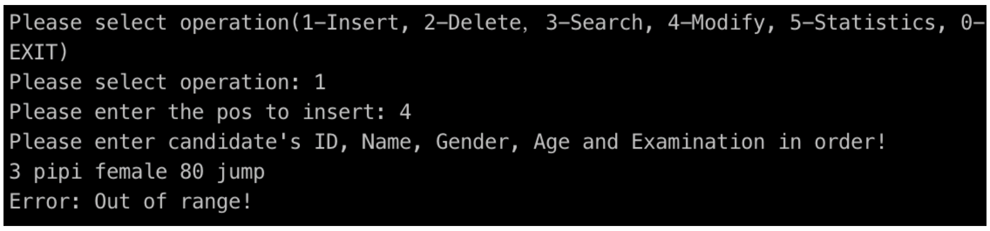
4.3.3 No such insertion place

No such insertion place

Expected Result:

The program runs normally without crashing and print ”Error: Out of range!”

Program Result:



```
Please select operation(1-Insert, 2-Delete, 3-Search, 4-Modify, 5-Statistics, 0-
EXIT)
Please select operation: 1
Please enter the pos to insert: 4
Please enter candidate's ID, Name, Gender, Age and Examination in order!
3 pipi female 80 jump
Error: Out of range!
```

Figure 4.12: Screenshots

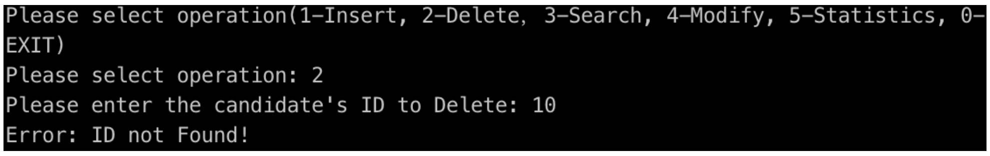
4.3.4 No such ID

No such ID

Expected Result:

The program runs normally without crashing and print ”Error: ID not Found!”

Program Result:



```
Please select operation(1—Insert, 2—Delete, 3—Search, 4—Modify, 5—Statistics, 0—
EXIT)
Please select operation: 2
Please enter the candidate's ID to Delete: 10
Error: ID not Found!
```

Figure 4.13: Screenshots

Appendix A

Frequently Asked Questions

A.1 Why this document looks so sparse?

This document is built using LaTeX and is arranged in book format. There may be blank pages before and after each chapter.

A.2 Why is there header files in this project?

I wrote basic stl header files, such as vector.h, etc. Except for the header files I wrote, the only header files used in all projects are iostream and string.h