CA Report

Demonstration of functionality

1. Brief introduction

In general, our code achieves a rough, small data processing code to simulate an appointment system which allows hospital to deal with mass medical treatment problem.

1.1. Data Structure

There are several data structure in the whole project: list: Listext and Fibonacci heap: FibHeap and FibNode,database, B Tree and B+ Tree.

1.1.1. list

Actually, we reference the code from class to build list data structure Listext, so you can find the similar code from lab4 assignment. However, in order to better fit our own situation, not only do we modify the index in pointer Reprarray from 1 to 0, but also we add more member functions in list structure Listext so that we can achieve the interaction between list structure and Fibonacci heap structure.

1.1.2. Fibonacci heap

We use classical Fibonacci heap to satisfy the requirement of sorting. However, we carefully designed key value. Taken job first for example,to be specific, the key value has six figures. From left to right, the first figure record job grades, and the second figure record age grades. And the third, forth and fifth figures record the dates. Then for the last figure, “0” represents morning, and “1” represents afternoon. And this time we build three heaps to achieve the goals to meet the challenge three types of treatment.

1.1.3. Database & block

The main function of the database is to store all the personal informations. We designed a relational database schema with four different relations Person, Medical Status,Registration and Treatment capturing all data that are to be collected during registration, scheduling and treatment processing. We almost all the useful important functions in this class. We can insert, delete and retrieve both a specific person provided with his id and block id, and a specific block.

We have different priority rules this time. The priorities are listed as followed:

1). job first

2). age first

3). appointment time first

We also represent each of these relations in a data structure that uses an ordered sequence of blocks, where each block is realised by an array. And we can find the corresponding block by searching for its block id.

1.1.4. B Tree/ B+ Tree

In both trees, we define the order in the front of the codes. The function of the trees is searching, which receive a person's ID and return the block id which refers to where the personal information is stored. After finding the address, we can take other operations later.

For B Tree, we use the open source codes from https://blog.csdn.net/yantingtao_sunny/article/details/51231934. The source code is powerful, but it cannot satisfies our needs. So we set up a new class called node to substitude the template class T. The class node contains key value and an array which is used to store the block ids of the people who has that key value. We also revised other fundamental functions like insert, delete and so on.

```
template <class T>
class BTnode
{
  friend class BTree<T>;//友元函数
public:
    BTnode();
    void inserts(T values);//在已有key的list中插入blockid
    int deletes(T values);//在已有key的list中删除blockid，返回值为list中元素数量，若数量为零，不用再删除
    Listext<int>* finds(T values);//返回key对应的list的指针
    bool isleaf;//判断节点是否是叶子节点
    int keyNum;//关键字个数
    BTnode<T>* parent;//指向父节点
    BTnode<T>* pchild[child_max];//子树指针数组
    T   keyvalue[key_max];//关键字数组
};
```

And also, we define the overload function <=, = and etc to keep the code working.

```
class node
{
public:
    node()
    {
        key=-1;
        blockid=new Listext<int>();
    }
    node(int k, int id);
    int key;
    Listext<int> *blockid;
    void insertid(int id);
    int deleteid(int id);
    bool operator== (node node1)
    {
        return key==node1.key;
    }
    bool operator<= (node node1)
    {
        return key<=node1.key;
    }
    bool operator>= (node node1)
    {
        return key>=node1.key;
    }
    bool operator< (node node1)
    {
        return key<node1.key;
    }
    bool operator> (node node1)
    {
        return key>node1.key;
    }

};
```

For B+ Tree, we search the open source code from https://blog.csdn.net/liu1064782986/article/details/7982290. In order to achieve this function and build the tree, we redefine some functions.

```
// 结点基类
class CNode{
public:
    CNode();
    virtual ~CNode();

    NODE_TYPE getType() const {return m_Type;}
    void setType(NODE_TYPE type){m_Type = type;}
    int getKeyNum() const {return m_KeyNum;}
    void setKeyNum(int n){m_KeyNum = n;}
    KeyType getKeyValue(int i) const {return m_KeyValues[i];}
    void setKeyValue(int i, KeyType key){m_KeyValues[i] = key;}
    int getKeyIndex(KeyType key)const;  // 找到键值在结点中存储的下标
    // 纯虚函数，定义接口
    virtual void removeKey(int keyIndex, int childIndex)=0;  // 从结点中移除键值
    virtual void split(CNode* parentNode, int childIndex)=0; // 分裂结点
    virtual void mergeChild(CNode* parentNode, CNode* childNode, int keyIndex)=0;  // 合并结点
    virtual void clear()=0; // 清空结点，同时会清空结点所包含的子树结点
    virtual void borrowFrom(CNode* destNode, CNode* parentNode, int keyIndex, SIBLING_DIRECTION d)=0; // 从兄弟结点中借一个键值
    virtual int getChildIndex(KeyType key, int keyIndex)const=0;  // 根据键值获取孩子结点指针下标
protected:
    NODE_TYPE m_Type;
    int m_KeyNum;
    KeyType m_KeyValues[MAXNUM_KEY];
};
```

```
// 内结点
class CInternalNode : public CNode{
public:
    CInternalNode();
    virtual ~CInternalNode();

    CNode* getChild(int i) const {return m_Childs[i];}
    void setChild(int i, CNode* child){m_Childs[i] = child;}
    void insert(int keyIndex, int childIndex, KeyType key, CNode* childNode);
    virtual void split(CNode* parentNode, int childIndex);
    virtual void mergeChild(CNode* parentNode, CNode* childNode, int keyIndex);
    virtual void removeKey(int keyIndex, int childIndex);
    virtual void clear();
    virtual void borrowFrom(CNode* destNode, CNode* parentNode, int keyIndex, SIBLING_DIRECTION d);
    virtual int getChildIndex(KeyType key, int keyIndex)const;
private:
    CNode* m_Childs[MAXNUM_CHILD];
};
```

```
// 叶子结点
class CLeafNode : public CNode{
public:
    CLeafNode();
    virtual ~CLeafNode();

    CLeafNode* getLeftSibling() const {return m_LeftSibling;}
    void setLeftSibling(CLeafNode* node){m_LeftSibling = node;}
    CLeafNode* getRightSibling() const {return m_RightSibling;}
    void setRightSibling(CLeafNode* node){m_RightSibling = node;}
    DataType getData(int i) const {return m_Datas[i];}
    void setData(int i, const DataType& data){m_Datas[i] = data;}
    void insert(KeyType key, const DataType& data);
    virtual void split(CNode* parentNode, int childIndex);
    virtual void mergeChild(CNode* parentNode, CNode* childNode, int keyIndex);
    virtual void removeKey(int keyIndex, int childIndex);
    virtual void clear();
    virtual void borrowFrom(CNode* destNode, CNode* parentNode, int keyIndex, SIBLING_DIRECTION d);
    virtual int getChildIndex(KeyType key, int keyIndex)const;
private:
    CLeafNode* m_LeftSibling;
    CLeafNode* m_RightSibling;
    DataType m_Datas[MAXNUM_LEAF];
};
#endif
```

```
class basetype
{
private:
    /* data */
public:
    basetype(/* args */);
    basetype( person_info* per ){
        ID = per->ID;
        bedelete = false;
        blockID = -1;
        countin = -1;
        countout = -1;
        betreated = false;
        waiting_time = 0;
    }
```

```
template<class T> class block
{
private:
    /* data */
public:
    block(/* args */);
    ~block();
    int block_ID;
    Listext<T>* reparray;
    Listext<T>* overflowbuffer;
    int maxsize;
    int overflowbig;
    int overflowsmall;
    int size;
    bool isfull();
    bool isempty();
    void insert(T elem);
    void sort();
    void remove(T elem);
    void remove(int ID);
    T retrieve(int ID);
    T binarysearch(int ID);
    T ordsearch(int ID);
    void overtorep();
    void reptovoer();
    void updateday(int counter);
};
```

2. Input and initialize

2.1. Primary time input and hospital settings:

Based on our test file, the default date of this code is 2022.4.1, and the default settings of **hospital** is ready if run the code directly. We also leave the code to test other choices if needed, which is left in the comment, just like below:

```
/* cout << "year:" << endl;
 cin>> choose1;
 year = stoi(choose1);
 cout << "month:" << endl;
 cin>> choose1;
 month = stoi(choose1);
 cout << "day:" << endl;
 cin>> choose1;
 day = stoi(choose1);
 cout << "hospital 1  address_x:" << endl;
 cin>> choose1;
 h1x = stoi(choose1);
 cout << "hospital 1  address_y:" << endl;
 cin>> choose1;
 h1y = stoi(choose1);
 cout << "hospital 1  capacity_x:" << endl;
 cin>> choose1;
 h1c = stoi(choose1);
 cout << "hospital 2  address_x:" << endl;
 cin>> choose1;
```

2.2. Choose interface

When entering the code, the below words will be printed in the screen:

```
please input the primary time
input 1 to assess the system
input 2 to input file of seven day
input 3 to input file of one month
input 4 to quit
```

If choose 2 or 3, personal information of one week or one month will be inputed into the database automatically.

If choose to enter 1, you will see as below :

```
hello! Who are you
If you are staff and want to input the file, please input 1
If you are patient, please input 2
If you want to quit the system, please input 3!
```

(1) staff

For the choice above, if you input 1, personal information for half a day will be inputed into the database.

(2) patient

```
If you are staff and want to input the file, please input 1
If you are patient, please input 2
If you want to quit the system, please input 3!
2
```

For the choice appeared above, if you choose 2, you will see the below information:

```
please choose the business
if you want to withdraw registion, please input 1!
if you want to change your information, input 2!
```

a. Withdraw

If you choose to withdraw, you will see the below choice:

```
input which treatment you want to withdraw
please input 1 to choose treatment_1
please input 2 to choose treatment_2
please input 3 to choose treatment_3
```

This choice is to ask you which treatment you are going to withdraw from.

After choose one of these three, you will see the below:

```
input your ID number !
```

If the patient is not in registration or treatments, you can see.

```
input your ID number !
10017
bad
You never registered or have been treated, and you can not withdraw!
good bye!
```

If the patient is in registration or treatment lists,

```
input your ID number !
10021
withdraw successfully
good bye!
```

b. Update

input ID

```
if you want to change your information, input 2!
2
input your ID number !
10021
```

If the patient is not in the registration or treatment lists,

```
input your ID number !
10060
bad
You never registered!
good bye!
```

If the patient is in the registration or treatment lists,

```
input the information you want to change
if you changed job, please input 1 !
if your risk level changed, please input 2 !
if both are changed, please input 3 !
```

i. change job

```
input your new job grade!
2
update successfully!
good bye!
```

ii. change risk level

```
input your new risk grade!
0
No need to update!
update successfully!
good bye!
```

iii. change job and risk level

```
if both are changed, please input 3 !
3
input your new job grade!
0
input your new risk grade!
1
No need to update!
update successfully!
good bye!
```

*If you want to achieve the register again, you need to change the input file by yourself and withdraw it before you push the patient information in the second time.

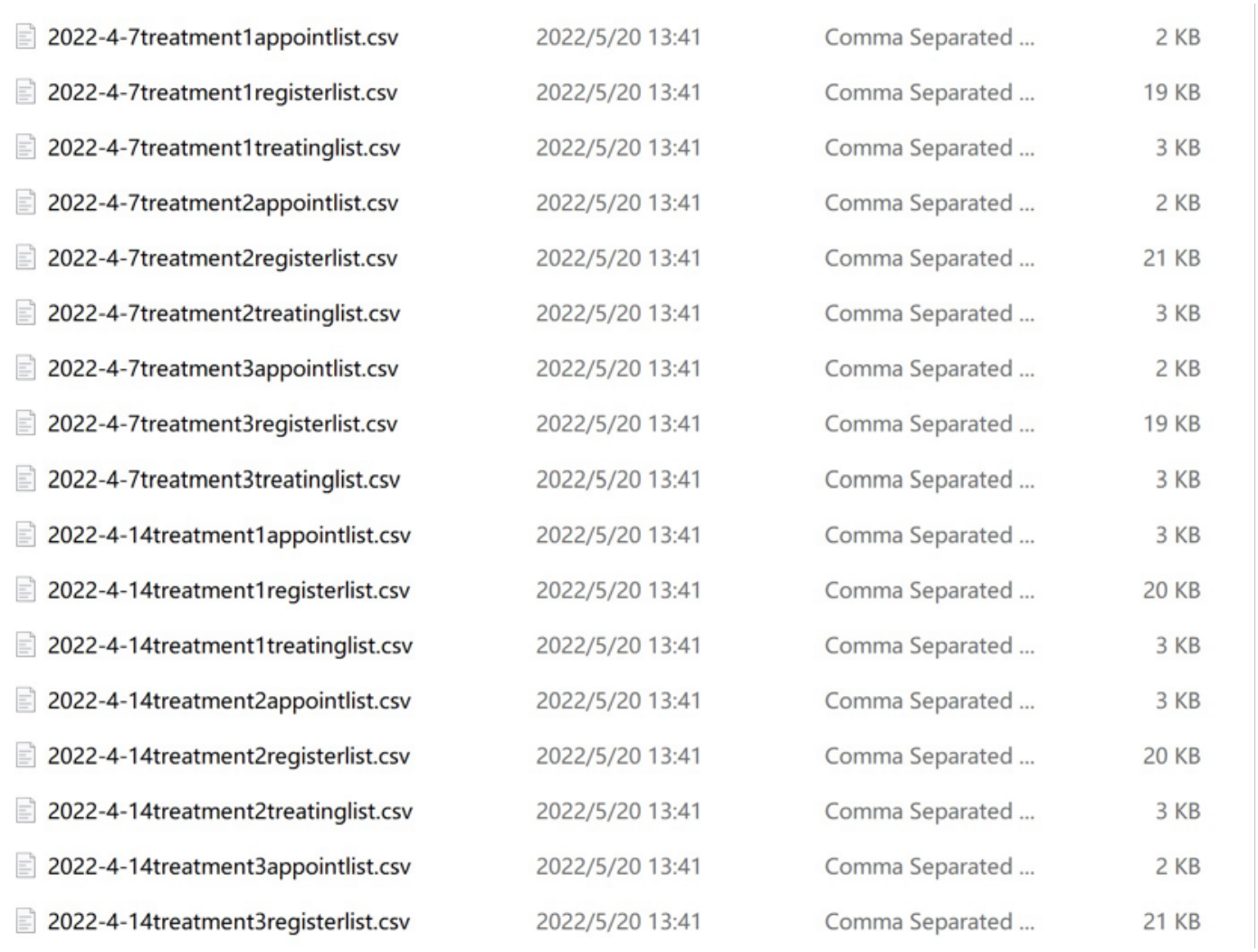
*In choose interface, you can enter 3 to quit this system.

```
If you are staff and want to input the file, please input 1
If you are patient, please input 2
If you want to quit the system, please input 3!
3
PS C:\Users\Lenovo\Desktop\CS225\Computing_Assignment1\good>
```

3. Output

3.1. Weekly report

Every week, after choosing to generate weekly report, the will create three file in xls format. And the file name will be date+list_name. The output file will be shown in the following.



The system shall produce weekly reports.

Appoint list

For appointment list, we will print the name, ID, telephone, register year, register month, register day, job grade, age grade, risk grade and waiting time. Here is an example sorting by name.

	A	B	C	D	E	F	G	H	I	J	K
1	name	ID	telephone	register ye	register m	register da	job_grade	age_grade	risk_grade	waiting_time	
2	ANYGTTVI	115035	2691799507	2022	4	7	2	1	1	1	
3	ATFTJFRIR	104947	526937946	2022	4	6	0	4	0	1	
4	AXJESGZA	105827	1037411598	2022	4	3	4	2	0	4	
5	BDCMWV	113883	4198372285	2022	4	6	4	3	0	2	
6	BOYOAQS	107801	3349247579	2022	4	7	1	3	0	1	
7	BQFSIUFF	111199	1324013051	2022	4	7	2	1	1	1	
8	CFVWLQR	107598	4198823018	2022	4	6	3	4	0	1	
9	CSYABTNC	113738	4211814807	2022	4	7	0	5	0	1	
10	CXMPMTR	115714	2117695312	2022	4	7	3	4	0	1	
11	DELXRNZC	112922	696280877	2022	4	6	2	1	0	1	
12	DPXNIWPE	113394	1762262831	2022	4	7	2	3	0	0	
13	DXSOQLLJ	106647	3830979451	2022	4	7	1	6	0	1	
14	EWJVYTUZ	116909	3985932595	2022	4	7	2	3	1	1	
15	FONETTZII	110593	4276503031	2022	4	7	3	2	0	0	
16	GUSSRNR	105213	2552191549	2022	4	6	2	4	0	1	
17	HFOGLQZ	107851	2088165260	2022	4	4	4	4	0	4	
18	IUYEZDIJJJ	106265	1282981718	2022	4	7	3	4	0	0	
19	IZZKGLUIG	102313	54998887	2022	4	6	3	2	0	1	
20	JOIZQHLIN	103149	3452829098	2022	4	6	4	2	0	2	
21	KNVXNHQ	102531	1981559647	2022	4	6	2	2	1	1	
22	KRKPDOP	114356	417626778	2022	4	6	3	6	1	1	
23	LQRUILLQ	111025	1704167804	2022	4	6	0	2	0	1	
24	LVBFSCTV	103311	604767515	2022	4	6	0	1	1	1	

Treating list

For treatment list, we will print the ID, treat year, treat month, treat day and waiting time.

Here is an example.

	A	B	C	D	E	F
1	ID	treat year	treat mont	treat day	wait time	
2	117424	2022	4	2	1	
3	117358	2022	4	2	1	
4	116570	2022	4	2	1	
5	116289	2022	4	2	1	
6	116269	2022	4	2	2	
7	114530	2022	4	2	2	
8	114502	2022	4	2	2	
9	114442	2022	4	2	1	
10	113860	2022	4	2	2	
11	113407	2022	4	2	1	
12	112261	2022	4	2	1	
13	111235	2022	4	2	2	
14	108176	2022	4	2	1	
15	106160	2022	4	2	2	
16	104837	2022	4	2	2	
17	104663	2022	4	2	2	
18	102735	2022	4	2	1	
19	102191	2022	4	2	1	
20	101938	2022	4	2	1	
21	100718	2022	4	2	2	
22	100336	2022	4	2	1	
23	117878	2022	4	3	2	

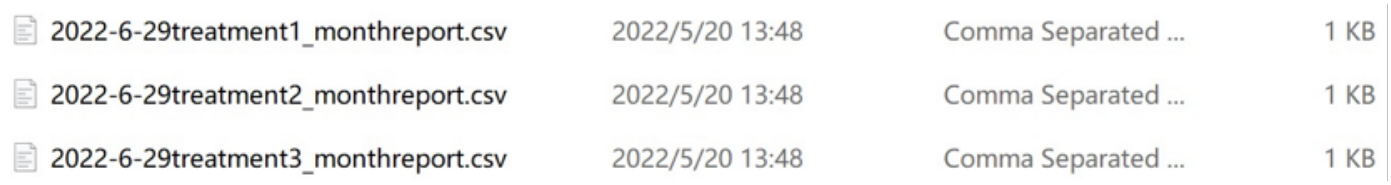
Register list

For Register list, we will print the name, ID, telephone, register year, register month, register day, job grade, age grade, risk grade and waiting time. Here is an example sorting by name.

	A	B	C	D	E	F	G	H	I	J	K
1	name	ID	telephone	register ye	register m	register da	job_grade	age_grade	risk_grade	waiting_time	
2	ABXNPULS	108016	722820076	2022	4	3	2	1	0	1	
3	ABZYYHH(	101958	723113705	2022	4	7	0	5	3	0	
4	ADFVSCDE	102311	1889705766	2022	4	7	3	2	2	0	
5	ADJQKMO	116799	2444232014	2022	4	7	6	2	2	0	
6	AEJWCYVL	114438	146992836	2022	4	4	3	2	3	3	
7	AMIIZFFAL	105260	2333830736	2022	4	5	2	2	2	2	
8	ANYGTTVI	115035	2691799507	2022	4	7	2	1	1	1	
9	AOPWWH	111242	3161437303	2022	4	5	2	3	0	1	
10	ATFTJFRIR(	104947	526937946	2022	4	6	0	4	0	1	
11	AWPUOST	104286	2951347078	2022	4	6	6	5	1	1	
12	AXCVFPBC	110097	2672092256	2022	4	1	7	6	0	6	
13	AXJESGZA	105827	1037411598	2022	4	3	4	2	0	4	
14	AXKHDAZI	107379	3478168518	2022	4	4	6	5	3	3	
15	BAZYNOQ	107916	3803363747	2022	4	6	4	4	0	1	
16	BBTKSSQX	116152	377448028	2022	4	3	0	5	0	1	
17	BDCMWV1	113883	4198372285	2022	4	6	4	3	0	2	
18	BEVZCMV:	107885	1054964003	2022	4	2	5	5	2	5	
19	BGVNQBH	104271	1981458239	2022	4	7	6	6	3	0	
20	BHLWOJQ	100270	1695640235	2022	4	1	2	2	2	6	
21	BJZWNEB(	114294	112099706	2022	4	4	5	2	3	3	
22	BMNSIXEC	109106	3596103563	2022	4	2	6	4	0	5	
23	BNZEZIXD	109975	3636076927	2022	4	2	1	2	1	1	

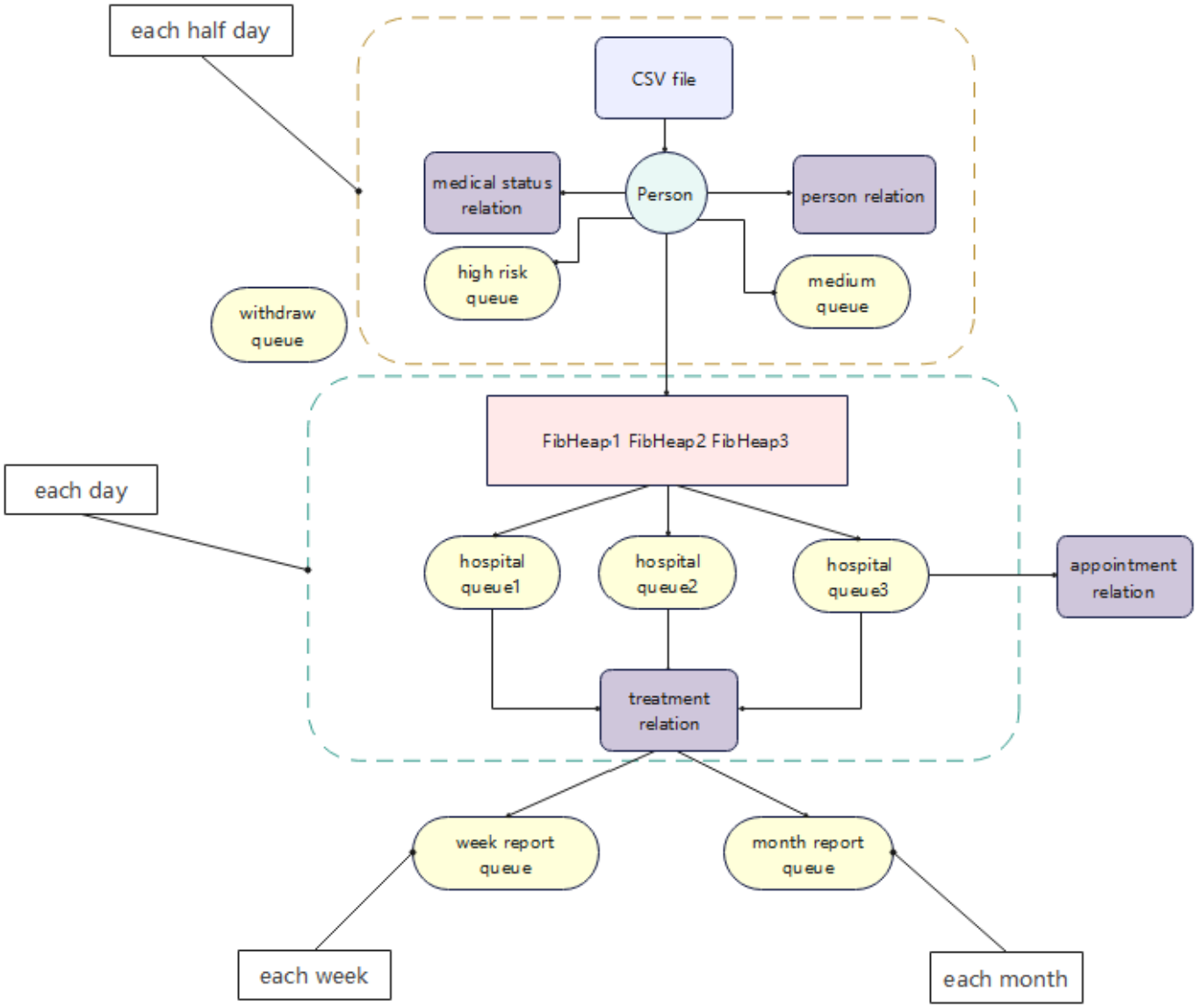
3.2. Monthly report

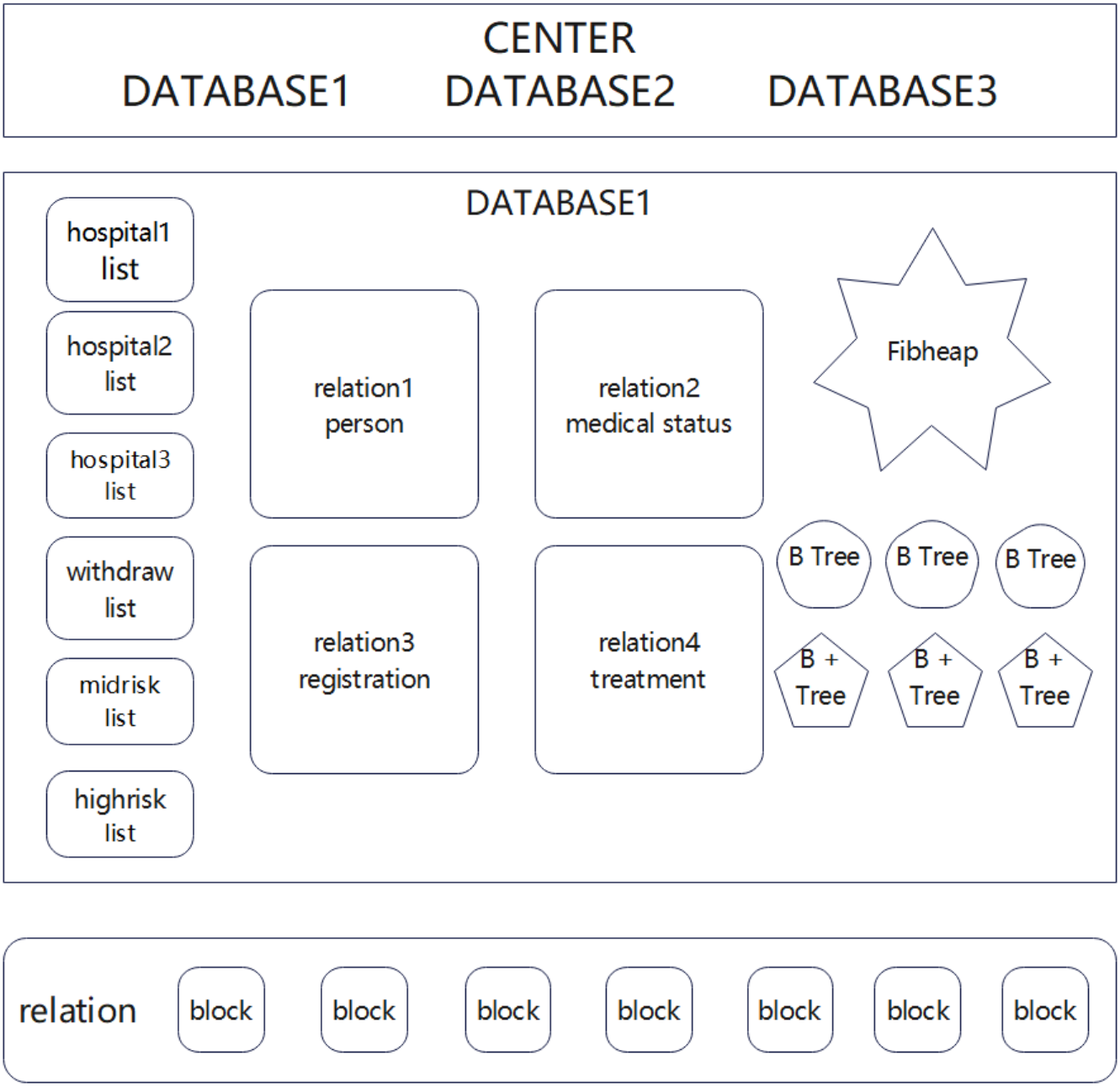
The system shall produce a monthly statistics report showing how many people have registered, how many of them are waiting, how many are waiting in total, how many treatment appointments have been made, the average waiting time, and the number of people who withdrew their registration. The file name is “date+monthly report”. A screen shot of the output file is shown below.



	A	B	C	D	E
1	total_people	now_wait_perople	treat_people	average_wait_time	withdraw_people
2	1963	1385	578	22	0

4. Flow chart


each half day
CSV file
medical status relation
Person
person relation
high risk queue
medium queue
withdraw queue
FibHeap1 FibHeap2 FibHeap3
each day
hospital queue1
hospital queue2
hospital queue3
appointment relation
treatment relation
week report queue
month report queue
each week
each month



Test

1. Registration

1.1. Integration of data from different local registries

We designed *local list* to store the *personal information* provided from different registries. After assembling the information, we put them into ***FibNode***. Next, we push the nodes into the ***FibHeap*** which is used as a centralized queue. In this way, we can achieve the goal of combining the data from different local registries.

2. Queueing and assignmnet of appoinments

2.1. Medium risk

In the class ***person_info*** there is a public value called ***risk_grade*** defined. When every ***person_info*** is popped out of the ***FibHeap***, before it is pushed to the appointment queue again, its ***risk_grade*** will be checked. Medium risk corresponds with ***risk_grade = 2***. When a medium risk is detected, the ***key_value*** of this ***person_info*** in ***FibHeap*** is added ***300***, which indicates that this person needs to wait for another 30 days in the priority queue. And then this person will be pushed back to the ***FibHeap*** again.

2.2. High risk

Similar from medium risk, high risk corresponds with ***risk_grade = 3***. When a high risk is detected, this ***person_info*** is popped out of the queue and added to a new list which is especially indicated for high risks. Only when the priority queue, which is the ***FibHeap***, is empty and the appointment queue is far from occupying the whole capacity of the hospitals, the high risks are pushed into the appointment queue to fill the whole capacity of the hospitals.

2.3. Update

When one person's imformation updated, this person's *person_info* will be inputted to the function, and this person will be searched by his or her *id*. According to the changes in the *person_info*, some relevant functions are implemented.

2.4. Withdraw

Every person has a value *date* in their own *person_info*. Before input one *person_info* into the heap, chech whether this person is in middle risk. If his or her risk grade is middle, add 60 to *date*, remove this person out of the heap and send this person to middle risk list. If this person wants to withdraw, add 14 to his or her *date*, remove him or her and move this person to withdraw list. Everyday, the *date* value of every person in middle risk list or withdraw list minus two. When this value equals to zero, put him or her out of this list and add to the heap.

3. Appointment processing

3.1. Distributed location

In the registration part, we have already collected the *address* of the patients which is represented as (x,y) and stored it in the ***FibNode***. In the *hospital* node, we also have the addresses of the hospitals. Therefore, we can calculate the distance and compare them. When we push the person with the highest priority, if he is not in the high risk or medium state, we distributed them to the nearest available hospital. And we add the pointer to the node which stored the personal information in the corresponding queues representing treatment points.

3.2. Appointment date and time

After assigning the patients into different hospitals, we set the appointed time of the patients and store it into the corresponding ***FibNode***.

3.3. Capacity

We have already store the ***capacity*** of the hospitals in the *hospital*. Everyday, the number of people assigned to hospitals is no more than the ***total capacity*** of the hospitals.

4. Reporting

4.1. Weekly report

4.1.1. Overall situation of the Week

The components of weekly report are treatment report, appointment report and waiting report.

- *Treatment report* is for those who have been treated. We print out their *name*, *ID*, *profession category*, *age category*, *risk status* and the *waiting time* from registration to treatment.
- Appointment report is for those who have an appointment but without treatment. Also, we print out their *name*, *ID*, *profession category*, *age category*, *risk status* and their *waiting time* until now.
- Waiting report is for those who even does not have an appointment. Again, we print out their *name, ID, profession category, age category, risk status* and their *waiting time* until now.

Where the data comes from:

- Get from input csv file: *name, ID, profession category, risk status.*
- Need to calculate:
    - *Age category*: What we get is the birth year, month* and *day* for a person. Therefore, according to change the specific date to *age category*, we calculate the distance between now and birth year.
    - *Waiting time*: Every time when we need to print the weekly report, we will calculate the waiting time for waiting and appointment people. And each time when a person finish treating, he or she will receive a waiting time filled in waiting_time parameter.

4.1.2. Ordered by name

We know that the type of name is *string*, so we can get each *char* in *string* and compare two *string*'s *chars* one by one. And the component of name is upper characteristics like "ZHANGSAN". We can directly compare the char sizes as in ASCII table, the upper characteristics are arranged from "A" to "Z" in x0041 to x005A.

Then, we use insert sort comparing and inserting people one by one.

4.1.3. Ordered by profession grade

This one is easy since that there are only 8 profession categories representing from 0 to 7. We simply use insert sort mentioned above to sort.

4.1.4. Ordered by age group

We use insert sort again. However, this time, in order to compare the *age date*, we compare *birth year* first, *birth month* second, and *birth day* third to decide the order, which means that we can not only sort by the *age category*, but also satisfy the actual age order.

4.2. Monthly report

In monthly report, we print the *total people, waiting people, treatment people, average waiting time* and *withdraw people* in this month.

- *Total people*: The total number of people registered.
- *Waiting people*: The number of people still wait for treating.
- *Treatment people*: The number of people finished the treatment.
- *Average waiting time*: Accumulate everyone's waiting time except withdraw people and calculate the average waiting time.
- *Withdraw people*: The number of people withdraw the process.