Instruction to Computing Assignment 2 Programs

Group 4

Kong Zitai	3190110306
Li Zihao	3190110098
Song Yifei	3190110099
Qi Zhenting	3190112155

Welcome to use our computing assignment 2 project and thanks for your patience to test and grade our work. To help you better test our programs, we write this instruction. You are welcome to check the codes and we highly recommend you to follow this instruction. The first 5 parts tell you how to use our program and the last part shows you the corresponding

I.Files

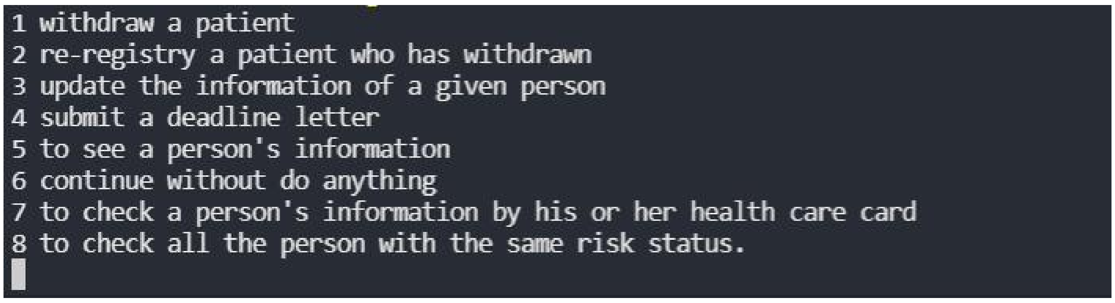
Together with this instruction, you can see a directory named *Computing 2 Group 4*, in the directory there are all the files we need to finish this project. Here are the lists:

Number	Type	Name
1	Source file 1	registry_1.csv
2	Source file 2	registry_2.csv
3	Source file 3	registry_3.csv
4	Source file 4	registry_4.csv
5	Source file 5	registry_5.csv
6	Head file	fibheap.h
7	Head file	report.h
8	Head file	person.h
9	Cpp file	main.cpp
10	Help head file	alist.h
11	Help head file	fifo.h
12	Help cpp file	alist.cpp
13	Help cpp file	fifo.cpp
14	Compile file	Makefile
15	Head file	B_Tree_Decl.h
16	Head file	B_Tree.h
17	Head file	BPlus_node.h
18	Head file	BPlus_tree.h
19	Head file	hashset.h
20	Cpp file	BPlus_node.cpp
21	Cpp file	BPlus_tree.cpp
22	Cpp file	hashset.cpp

II.Settings

Here are our settings of the system:

Our program just analogs a simplified real situation of a city's medicine system modified from Computing Assignment 1. We set 5 registries and 3 hospitals. The prepared input files are 5 registry_n.csv files, each file contains 500 people, each file represents the person that are registered in a registry. Each half day there will be 5 people per registry, 25 people in total, being plugged from the file to the queue, standing for that they were registered. And each hospital, in Alist form, can deal with 8 people per day. Each day, although the queues put people into Fibonacci heap for twice, the hospital only makes appointments for the next day by once. After they are "treated", they will be put into an Alist archive, B+ tree, B tree and hash table. At the process from queue to Fibonacci heap, and at the process from Fibonacci heap to hospital, we set two check points for you to excute the following operations:

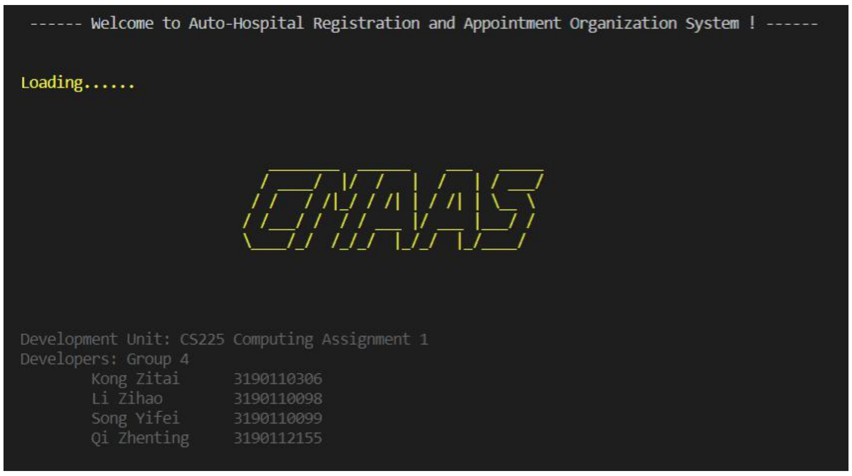


```
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
7 to check a person's information by his or her health care card
8 to check all the person with the same risk status.
```

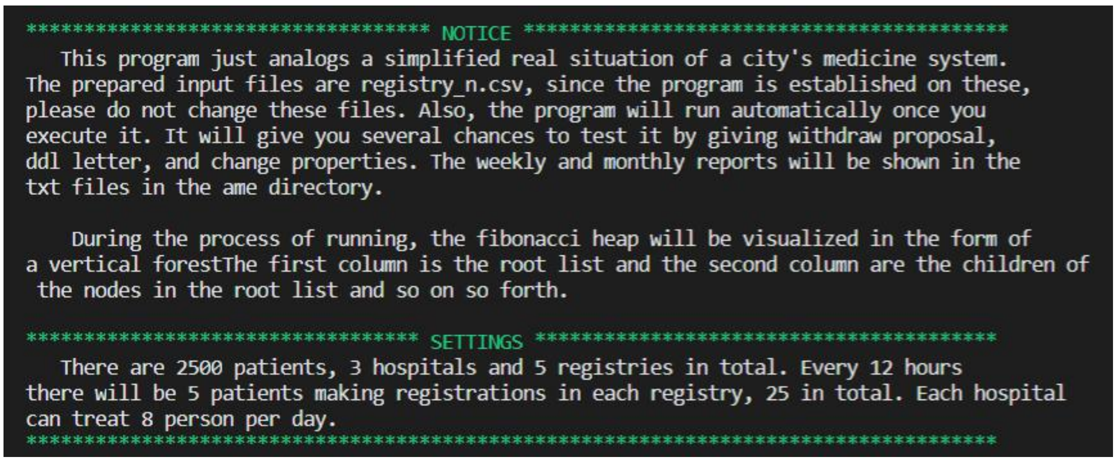
Each person will have the following artribute: id, registration, name, address, phone, Wechat, profession, birth, register_time, risk status, hospital, regist_halfday, treat_halfday, withdraw, withdraw_halfday, reinsert_halfday, risk2halfday, ddlletter, treatment, Euclidean_address, health_care_card; And these properties are stored into 4 relations which are in struct form. We established a B+ tree for each relation, thus there are 4 in total, and these B+ trees use id as the primary key. Also, we establish a B tree and a hash table to search for data based on secondary keys, health card number and risk status resp. When you enter the secondary key information, you can get the id and search for the corresponding information of those people with the secondary key in the B + tree. Here, to show these, we provide two additional kind of reports: week report based on the risk and month report based on the risk.

III.Execution Instruction

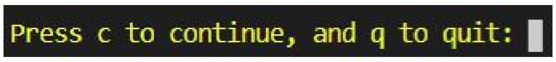
Please type "make" to compile. Then the executable is called "out.exe". Then type "./out" to execute the file and you will see the user interface. Once start the program will run automatically.



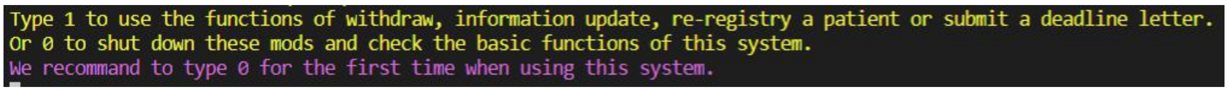
```
 ------ Welcome to Auto-Hospital Registration and Appointment Organization System ! ------

Loading......

Development Unit: CS225 Computing Assignment 1
Developers: Group 4
        Kong Zitai      3190110306
        Li Zihao        3190110098
        Song Yifei      3190110099
        Qi Zhenting     3190112155
```


```
*********************************** NOTICE ******************************************
   This program just analogs a simplified real situation of a city's medicine system.
The prepared input files are registry_n.csv, since the program is established on these,
please do not change these files. Also, the program will run automatically once you
execute it. It will give you several chances to test it by giving withdraw proposal,
ddl letter, and change properties. The weekly and monthly reports will be shown in the
txt files in the ame directory.

    During the process of running, the fibonacci heap will be visualized in the form of
a vertical forestThe first column is the root list and the second column are the children of
 the nodes in the root list and so on so forth.

*********************************** SETTINGS ****************************************
   There are 2500 patients, 3 hospitals and 5 registries in total. Every 12 hours
there will be 5 patients making registrations in each registry, 25 in total. Each hospital
can treat 8 person per day.
************************************************************************************
```

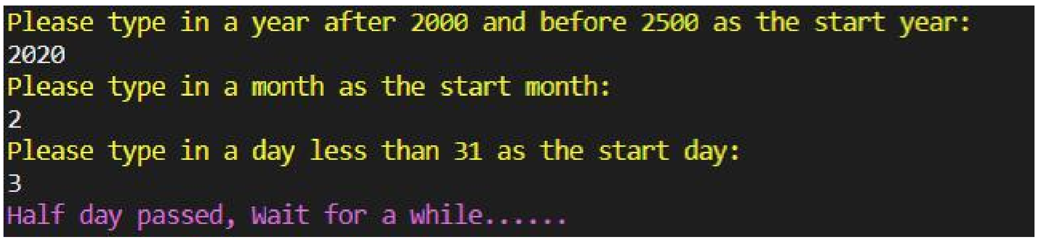
Now you may have a chance to decide whether to continue or not. If you want to stop, press "q". If you want to start, press "c".


```
Press c to continue, and q to quit:
```

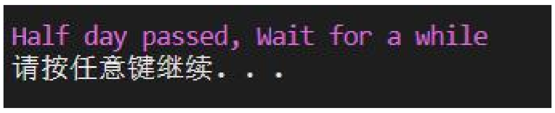
We set two modes: If you type 0 you will use it without a test window for withdraw, withdraw reinsert, information update and submit a deadline letter. If you type 1 you can test these. We highly recommend you to first type 0 to see the whole process of the program without these cases. Then use 1 mode to test these cases.


```
Type 1 to use the functions of withdraw, information update, re-registry a patient or submit a deadline letter.
Or 0 to shut down these mods and check the basic functions of this system.
We recommand to type 0 for the first time when using this system.
```

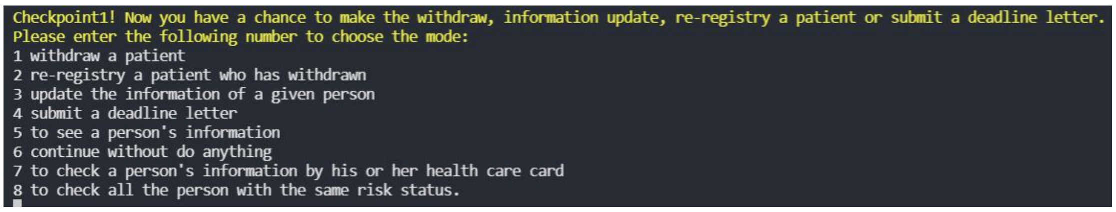
Now, please type in proper year/month/day to be the start time point.


```
Please type in a year after 2000 and before 2500 as the start year:
2020
Please type in a month as the start month:
2
Please type in a day less than 31 as the start day:
3
Half day passed, Wait for a while......
```

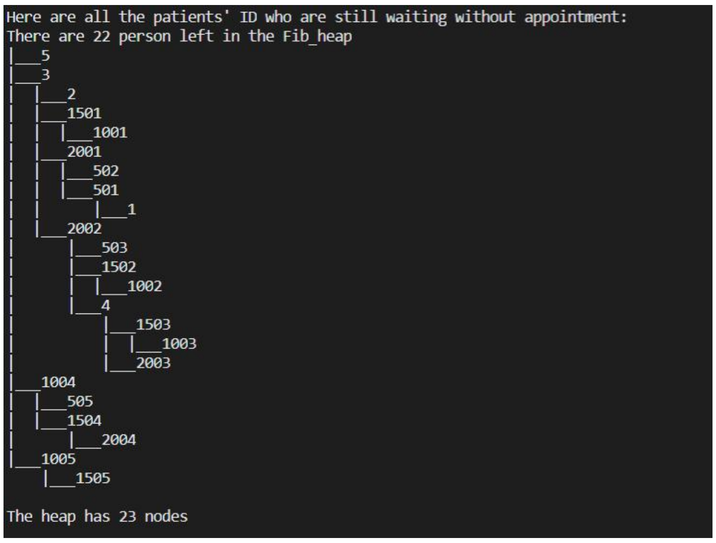
If you choose mode 0, there will be the following contents as the separator to mark that half day has passed. And you can stop to check the output information. To continue, press any key.


```
Half day passed, Wait for a while
请按任意键继续. . .
```

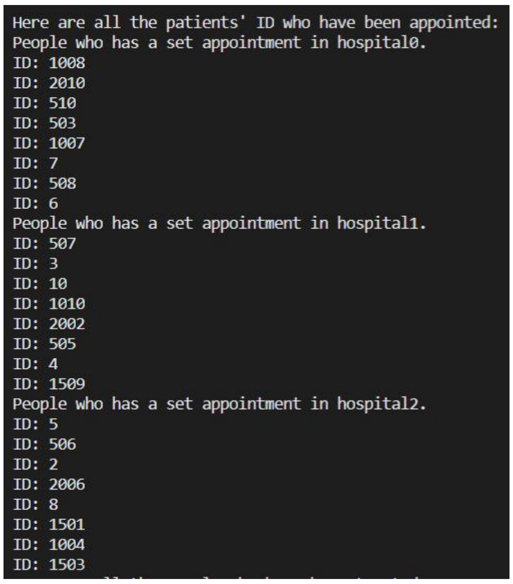
If you choose mode 1, there will be the following interface as the separator to mark that half day has passed. But there are two test points, the first is between the update of fibonacci heap and the hospital updates treated patient information and makes appointment; the second is after the hospital updates treated patient information and makes appointment. And you can press 1 to 6 to test. To continue, press 6.


```
Checkpoint1! Now you have a chance to make the withdraw, information update, re-registry a patient or submit a deadline letter.
Please enter the following number to choose the mode:
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
7 to check a person's information by his or her health care card
8 to check all the person with the same risk status.
```

For mode 1 to 6, First you will be shown the IDs you can operate, please choose one.

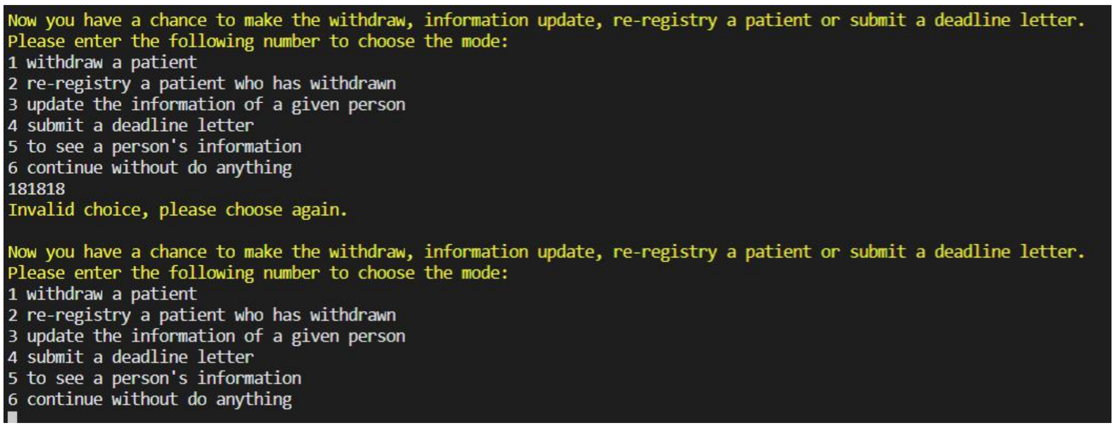


```
Here are all the patients' ID who are still waiting without appointment:
There are 22 person left in the Fib_heap
|___5
|___3
|  |___2
|  |___1501
|  |  |___1001
|  |___2001
|  |  |___502
|  |  |___501
|  |      |___1
|  |___2002
|      |___503
|      |___1502
|      |  |___1002
|      |___4
|          |___1503
|          |  |___1003
|          |___2003
|___1004
|  |___505
|  |___1504
|      |___2004
|___1005
    |___1505

The heap has 23 nodes
```

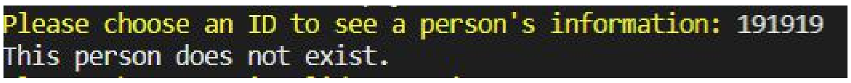


```
Here are all the patients' ID who have been appointed:
People who has a set appointment in hospital0.
ID: 1008
ID: 2010
ID: 510
ID: 503
ID: 1007
ID: 7
ID: 508
ID: 6
People who has a set appointment in hospital1.
ID: 507
ID: 3
ID: 10
ID: 1010
ID: 2002
ID: 505
ID: 4
ID: 1509
People who has a set appointment in hospital2.
ID: 5
ID: 506
ID: 2
ID: 2006
ID: 8
ID: 1501
ID: 1004
ID: 1503
```

If there is invalid information, you will receive the following tips to re-try.

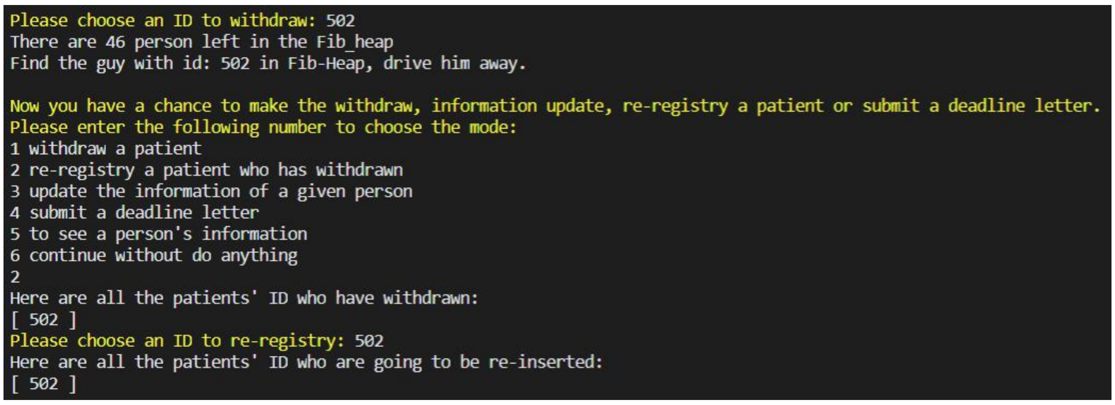


```
Now you have a chance to make the withdraw, information update, re-registry a patient or submit a deadline letter.
Please enter the following number to choose the mode:
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
181818
Invalid choice, please choose again.

Now you have a chance to make the withdraw, information update, re-registry a patient or submit a deadline letter.
Please enter the following number to choose the mode:
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
```



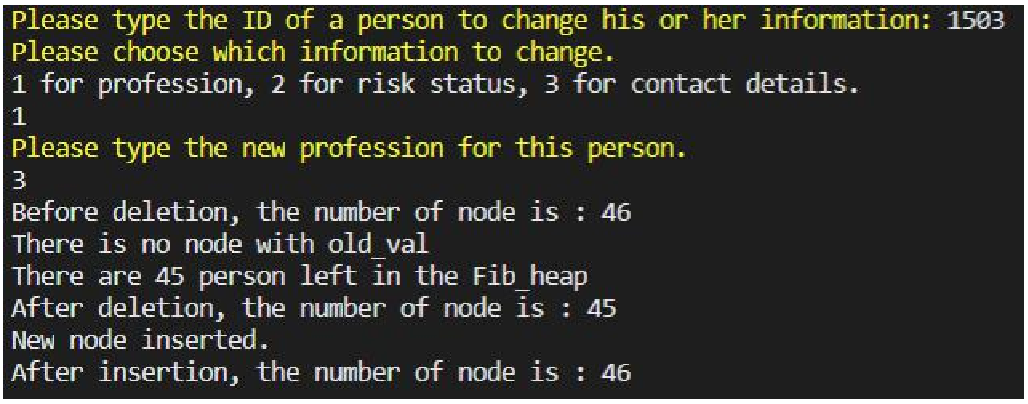
```
Please choose an ID to see a person's information: 191919
This person does not exist.
```

For 1, you can input the id to withdraw the person. Then you can use mode 2 to check because the mode 2 will print the list that for the withdrawn person. Then you can type this id to reinsert it. This id will be put into a wait list and wait here for 2 weeks.



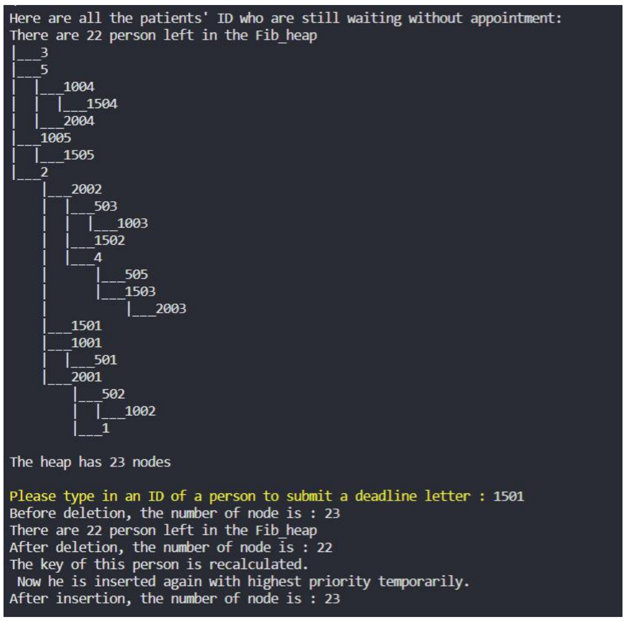
```
Please choose an ID to withdraw: 502
There are 46 person left in the Fib_heap
Find the guy with id: 502 in Fib-Heap, drive him away.

Now you have a chance to make the withdraw, information update, re-registry a patient or submit a deadline letter.
Please enter the following number to choose the mode:
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
2
Here are all the patients' ID who have withdrawn:
[ 502 ]
Please choose an ID to re-registry: 502
Here are all the patients' ID who are going to be re-inserted:
[ 502 ]
```

For 3, you can choose what to update. And the associated information will be shown for you to check.

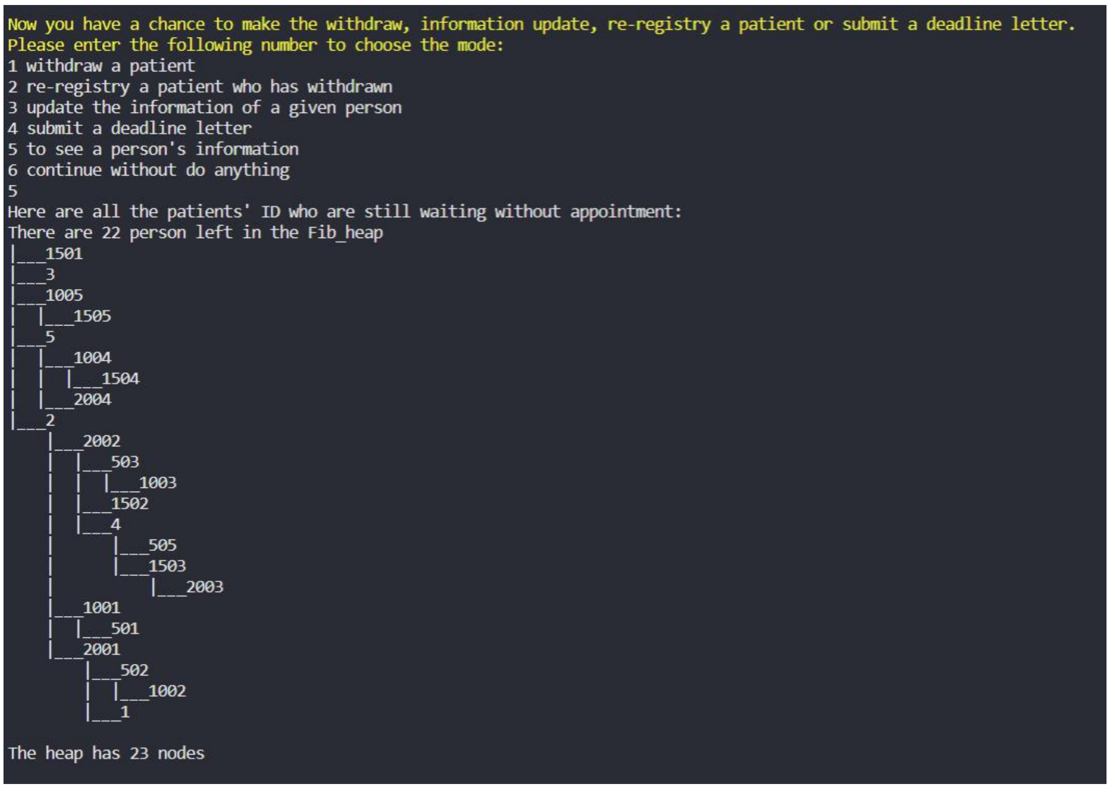


```
Please type the ID of a person to change his or her information: 1503
Please choose which information to change.
1 for profession, 2 for risk status, 3 for contact details.
1
Please type the new profession for this person.
3
Before deletion, the number of node is : 46
There is no node with old_val
There are 45 person left in the Fib_heap
After deletion, the number of node is : 45
New node inserted.
After insertion, the number of node is : 46
```

For 4, you first type in an id from the list, then the person will be given a deadline letter to make it the root of the Fibonacci heap, which will be appointed to the hospital as soon as possible.

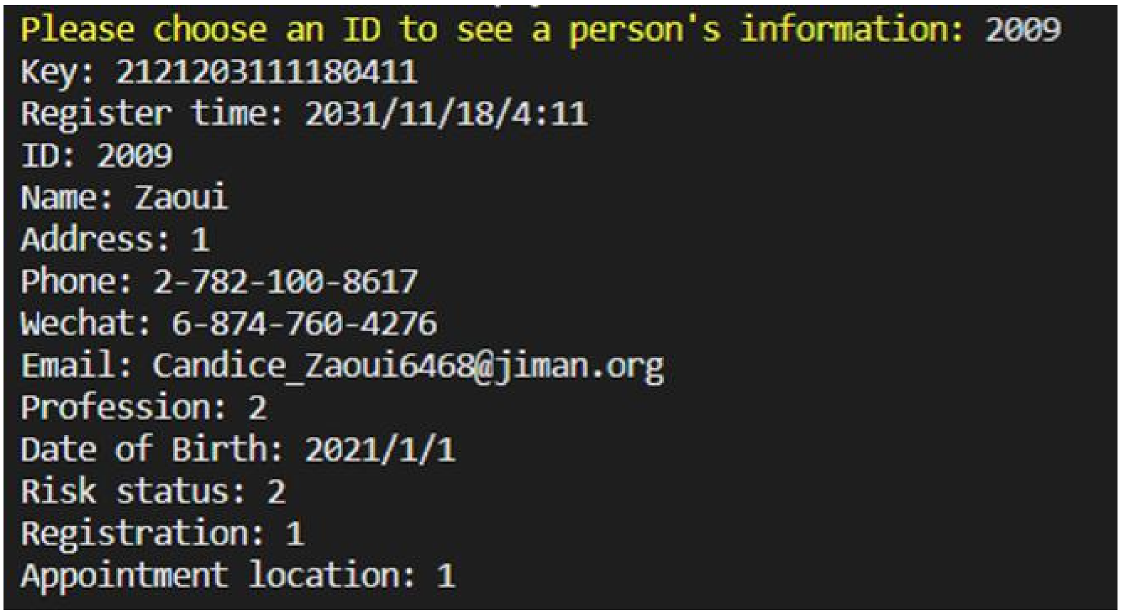


```
Here are all the patients' ID who are still waiting without appointment:
There are 22 person left in the Fib_heap
|___3
|___5
|  |___1004
|  |  |___1504
|  |___2004
|___1005
|  |___1505
|___2
    |___2002
    |  |___503
    |  |  |___1003
    |  |___1502
    |  |___4
    |      |___505
    |      |___1503
    |          |___2003
    |___1501
    |___1001
    |  |___501
    |___2001
        |___502
        |  |___1002
        |___1

The heap has 23 nodes

Please type in an ID of a person to submit a deadline letter : 1501
Before deletion, the number of node is : 23
There are 22 person left in the Fib_heap
After deletion, the number of node is : 22
The key of this person is recalculated.
 Now he is inserted again with highest priority temporarily.
After insertion, the number of node is : 23
```



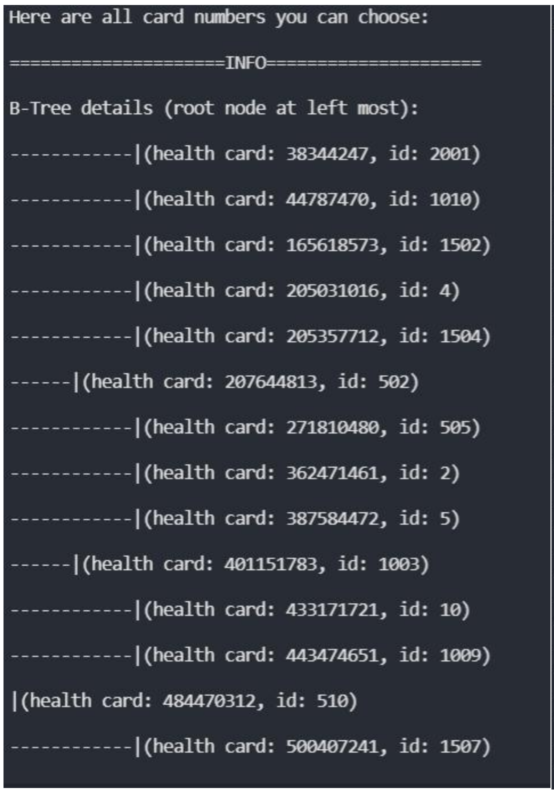
```
Now you have a chance to make the withdraw, information update, re-registry a patient or submit a deadline letter.
Please enter the following number to choose the mode:
1 withdraw a patient
2 re-registry a patient who has withdrawn
3 update the information of a given person
4 submit a deadline letter
5 to see a person's information
6 continue without do anything
5
Here are all the patients' ID who are still waiting without appointment:
There are 22 person left in the Fib_heap
|___1501
|___3
|___1005
|   |___1505
|___5
|   |___1004
|   |   |___1504
|   |___2004
|___2
     |___2002
     |   |___503
     |   |   |___1003
     |   |___1502
     |   |___4
     |       |___505
     |       |___1503
     |           |___2003
     |___1001
     |   |___501
     |___2001
         |___502
         |   |___1002
         |___1

The heap has 23 nodes
```

For 5, it will show you the detailed information of all the states of the system, from which you can choose an id of a person to show his detailed information. Please notice that at check point 1, what is shown are the people who are treated the day before so the appointment list should be empty, and at check point 2 what is shown are the people who are treated the day, so here there will be patients get appointed here. You can check by this useful tool.

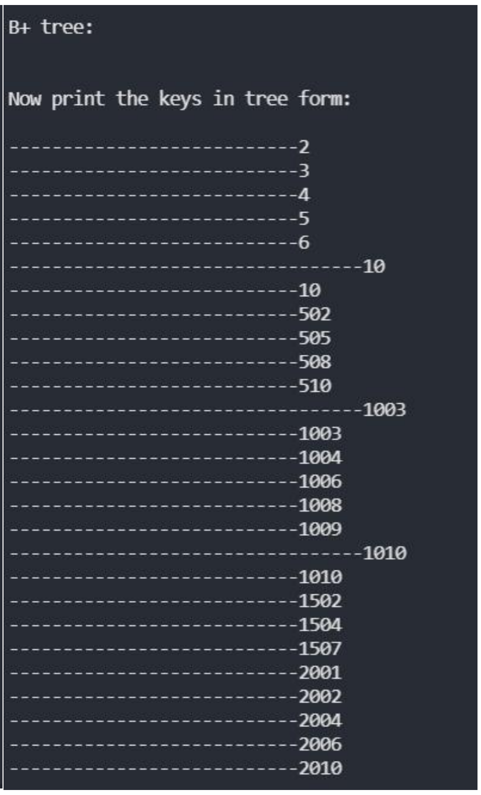


```
Please choose an ID to see a person's information: 2009
Key: 2121203111180411
Register time: 2031/11/18/4:11
ID: 2009
Name: Zaoui
Address: 1
Phone: 2-782-100-8617
Wechat: 6-874-760-4276
Email: Candice_Zaoui6468@jiman.org
Profession: 2
Date of Birth: 2021/1/1
Risk status: 2
Registration: 1
Appointment location: 1
```

For 7, it provides you a place to search information by B+ tree., but please first execute mode 6 for several times, because at the first time there is no people in B+ tree, nothing will be printed so the program will end. After several times, press 7, you will be shown what is in B+ and B tree, like,

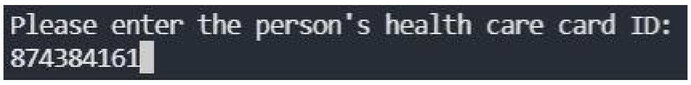


```
Here are all card numbers you can choose:

====================INFO====================

B-Tree details (root node at left most):

-----------|(health card: 38344247, id: 2001)

-----------|(health card: 44787470, id: 1010)

-----------|(health card: 165618573, id: 1502)

-----------|(health card: 205031016, id: 4)

-----------|(health card: 205357712, id: 1504)

------|(health card: 207644813, id: 502)

-----------|(health card: 271810480, id: 505)

-----------|(health card: 362471461, id: 2)

-----------|(health card: 387584472, id: 5)

------|(health card: 401151783, id: 1003)

-----------|(health card: 433171721, id: 10)

-----------|(health card: 443474651, id: 1009)

|(health card: 484470312, id: 510)

-----------|(health card: 500407241, id: 1507)
```



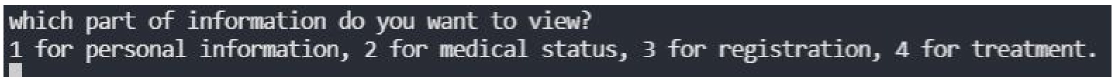
```
B+ tree:


Now print the keys in tree form:

---------------------------2
---------------------------3
---------------------------4
---------------------------5
---------------------------6
---------------------------------10
---------------------------10
---------------------------502
---------------------------505
---------------------------508
---------------------------510
---------------------------------1003
---------------------------1003
---------------------------1004
---------------------------1006
---------------------------1008
---------------------------1009
---------------------------------1010
---------------------------1010
---------------------------1502
---------------------------1504
---------------------------1507
---------------------------2001
---------------------------2002
---------------------------2004
---------------------------2006
---------------------------2010
```

And you can choose a health card number from above,



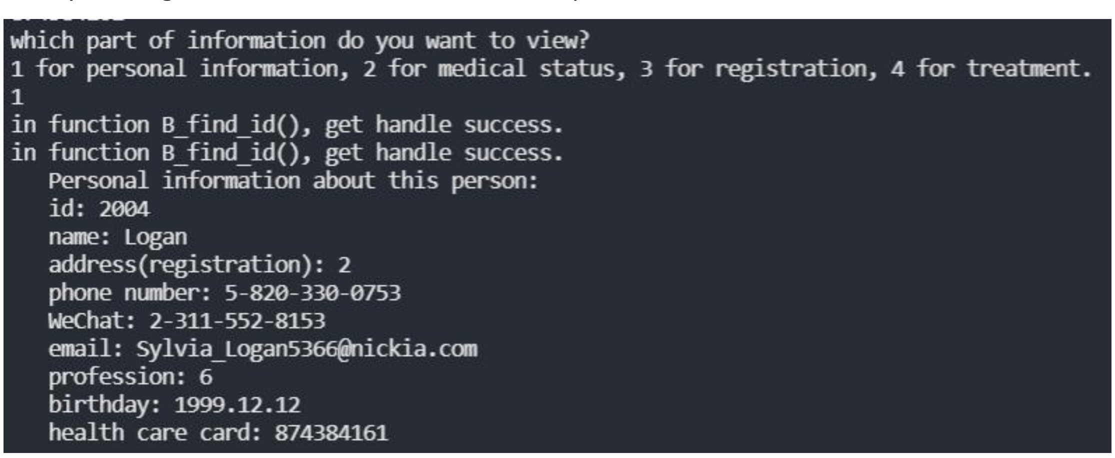
```
Please enter the person's health care card ID:
874384161
```

Then you can choose which relation you want to see.



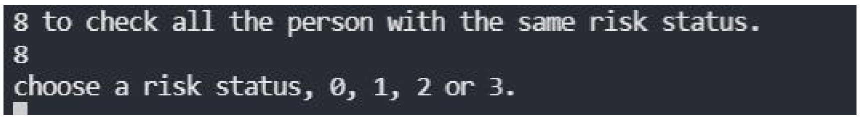
```
which part of information do you want to view?
1 for personal information, 2 for medical status, 3 for registration, 4 for treatment.
```

Then you can get all the aimed information of the person.



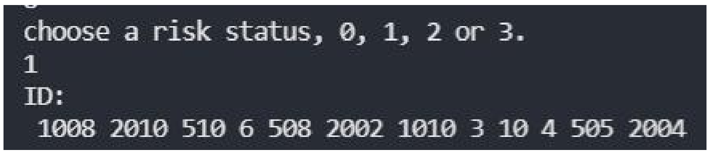
```
which part of information do you want to view?
1 for personal information, 2 for medical status, 3 for registration, 4 for treatment.
1
in function B_find_id(), get handle success.
in function B_find_id(), get handle success.
   Personal information about this person:
   id: 2004
   name: Logan
   address(registration): 2
   phone number: 5-820-330-0753
   WeChat: 2-311-552-8153
   email: Sylvia_Logan5366@nickia.com
   profession: 6
   birthday: 1999.12.12
   health care card: 874384161
```

For 8, it searches for different people with the same risk in B tree, and returns all the ids with that risk.



```
8 to check all the person with the same risk status.
8
choose a risk status, 0, 1, 2 or 3.
```

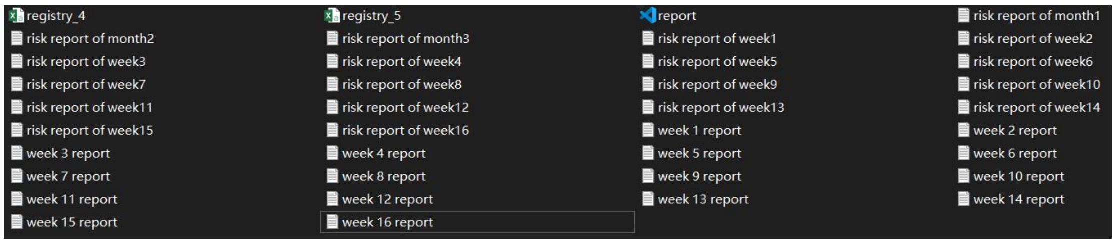
Then you can choose 1 to check.



```
choose a risk status, 0, 1, 2 or 3.
1
ID:
 1008 2010 510 6 508 2002 1010 3 10 4 505 2004
```

IV. Reports

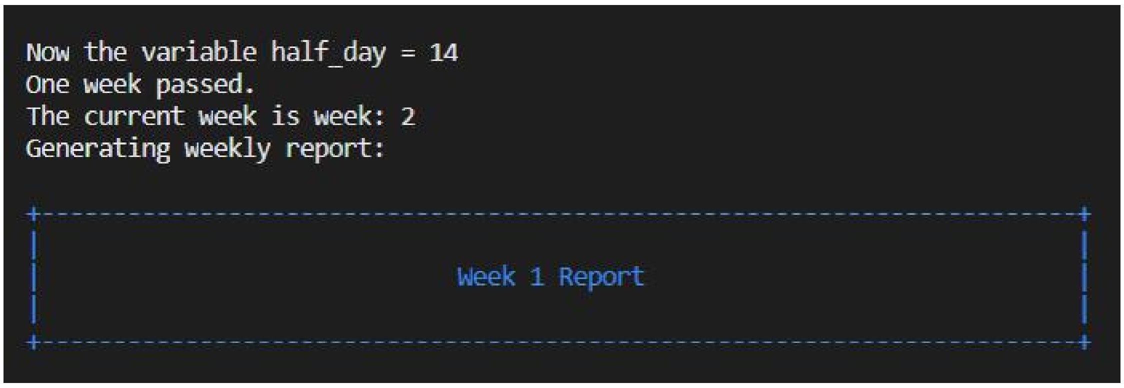
After execution, you can see the reports in the same directory as followings:



```
registry_4              registry_5              report                  risk report of month1
risk report of month2   risk report of month3   risk report of week1    risk report of week2
risk report of week3    risk report of week4    risk report of week5    risk report of week6
risk report of week7    risk report of week8    risk report of week9    risk report of week10
risk report of week11   risk report of week12   risk report of week13   risk report of week14
risk report of week15   risk report of week16   week 1 report           week 2 report
week 3 report           week 4 report           week 5 report           week 6 report
week 7 report           week 8 report           week 9 report           week 10 report
week 11 report          week 12 report          week 13 report          week 14 report
week 15 report          week 16 report
```

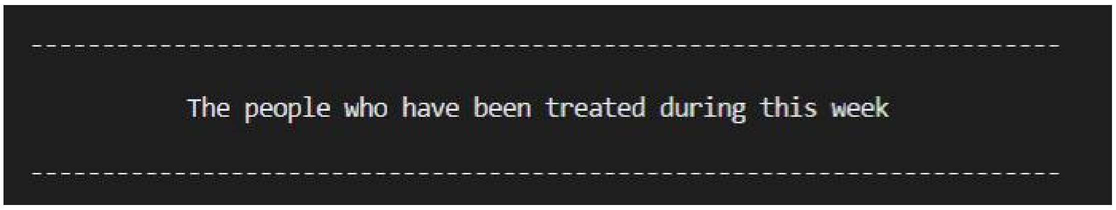
We provided 4 kinds of reports: week report, month report, risk report for week and risk report for month.

At the first day of each week, there will be a report of the last week. It has the head as:

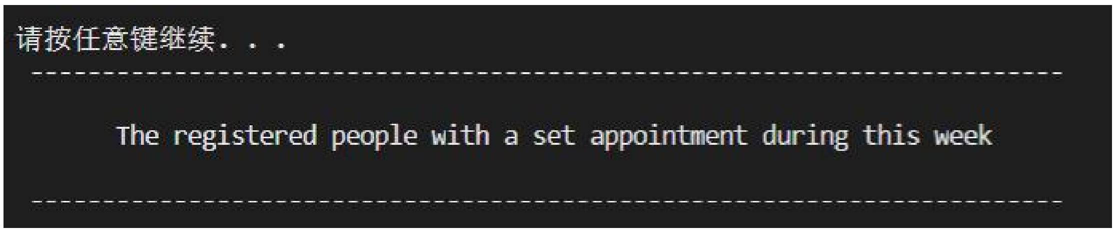


```
Now the variable half_day = 14
One week passed.
The current week is week: 2
Generating weekly report:

+------------------------------------------------------------------------+
|                                                                        |
|                           Week 1 Report                                |
|                                                                        |
+------------------------------------------------------------------------+
```

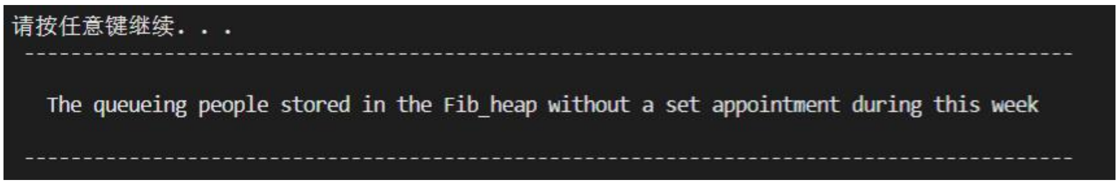
The fields are shown here:



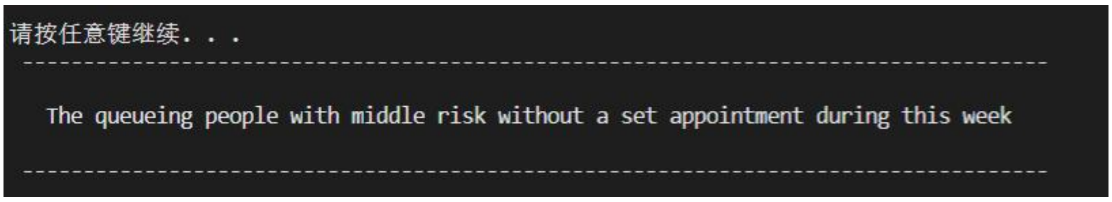
```
----------------------------------------------------------------------

          The people who have been treated during this week

----------------------------------------------------------------------
```



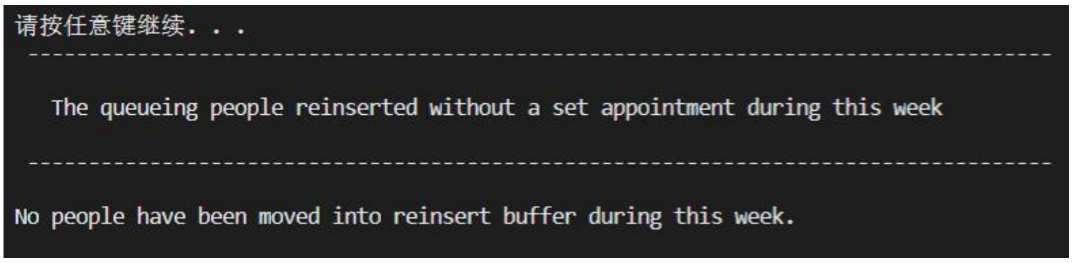
```
请按任意键继续. . .
----------------------------------------------------------------------

     The registered people with a set appointment during this week

----------------------------------------------------------------------
```



```
请按任意键继续. . .
--------------------------------------------------------------------------------

  The queueing people stored in the Fib_heap without a set appointment during this week

--------------------------------------------------------------------------------
```

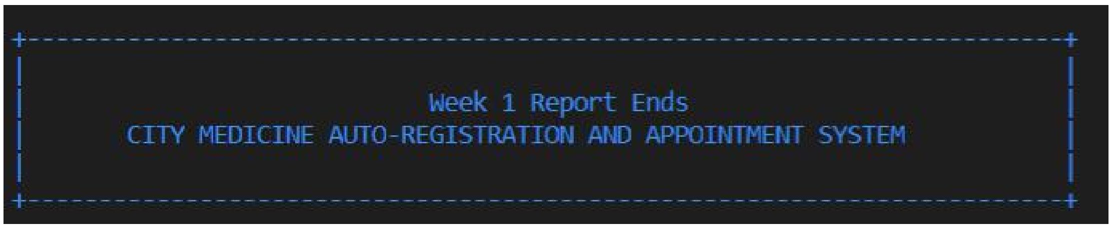


```
请按任意键继续. . .
 ---------------------------------------------------------------------------------

   The queueing people with middle risk without a set appointment during this week

 ---------------------------------------------------------------------------------
```



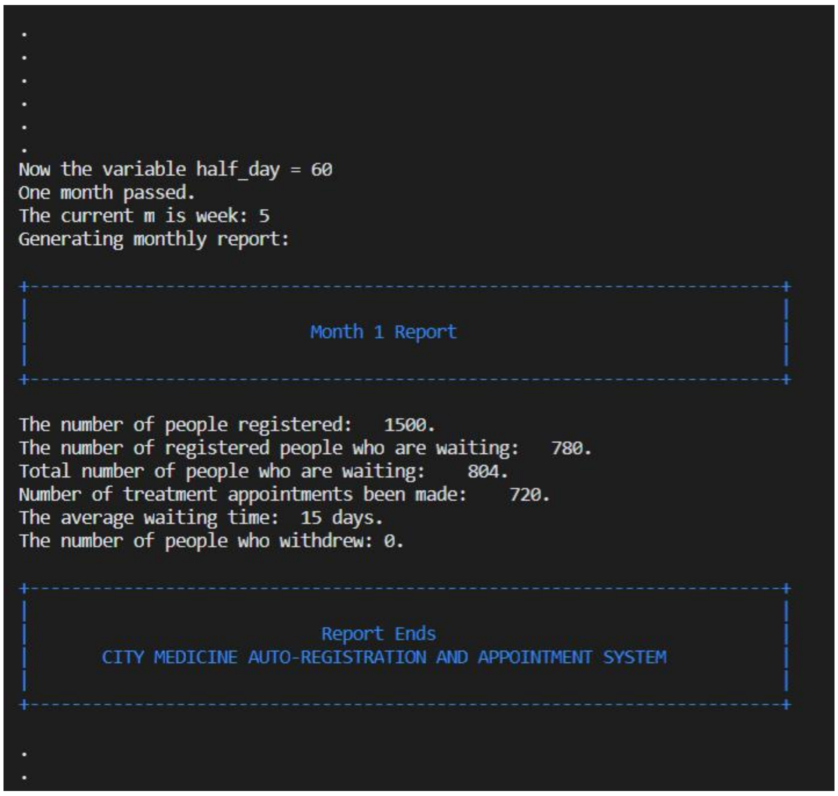
```
请按任意键继续. . .
 ---------------------------------------------------------------------------------

   The queueing people reinserted without a set appointment during this week

 ---------------------------------------------------------------------------------

No people have been moved into reinsert buffer during this week.
```

The end of the weekly report is like:



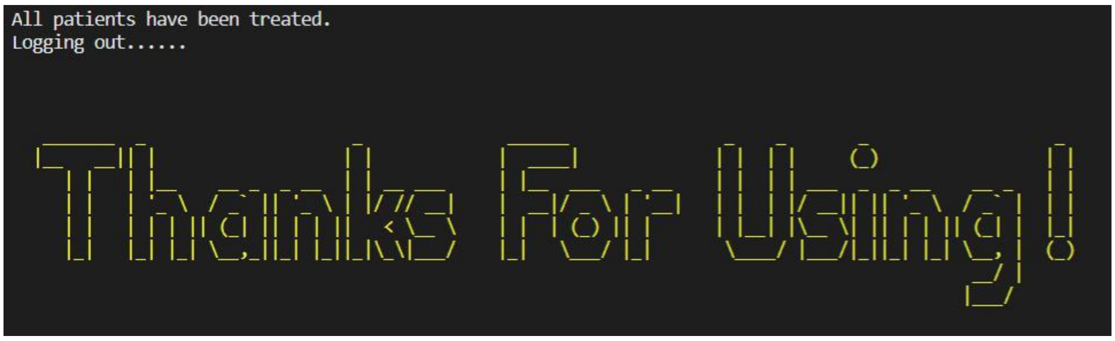
```
+---------------------------------------------------------------------------------+
|                                                                                 |
|                              Week 1 Report Ends                                 |
|         CITY MEDICINE AUTO-REGISTRATION AND APPOINTMENT SYSTEM                  |
|                                                                                 |
+---------------------------------------------------------------------------------+
```

At the first day of each month, there will be a report of the last month. It is like as:



```
.
.
.
.
.
.
Now the variable half_day = 60
One month passed.
The current m is week: 5
Generating monthly report:

+-------------------------------------------------------------------------+
|                                                                         |
|                              Month 1 Report                             |
|                                                                         |
+-------------------------------------------------------------------------+

The number of people registered:   1500.
The number of registered people who are waiting:   780.
Total number of people who are waiting:    804.
Number of treatment appointments been made:    720.
The average waiting time:  15 days.
The number of people who withdrew: 0.

+-------------------------------------------------------------------------+
|                                                                         |
|                              Report Ends                                |
|        CITY MEDICINE AUTO-REGISTRATION AND APPOINTMENT SYSTEM           |
|                                                                         |
+-------------------------------------------------------------------------+

.
.
```

When all the patients are treated, the program terminates. You will see this:



```
All patients have been treated.
Logging out......
```

However, you have to adjust the console size to see this full picture!

The whole program will take a while, about 16 weeks under the circumstance that you do not reinsert any withdrawn person.

Thanks and have fun!

V. Exercises

EXERCISE 1

(i) The design is that, the 4 relations are realized be struct in person.h, shown below

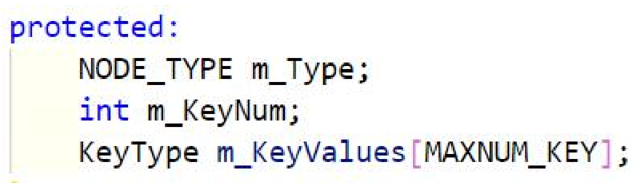
```
struct PERSON
{

    //some attributes
    int id;
    string name;                 //Name of the person
    string CONTACT_DETAIL[4];    //The array for storing the address, phone, WeChat and email of the person
    long long profession;        //The profession of the person(I to VIII)
                                 //1 stands for most important, 8 stands for least important
    long long birth[3];          //The date of birth of the person. The format is [year(yyyy),month(mm),day(dd)]
    int health_care_card;        //9 digits

};

struct MEDICAL_STATUS
{
    int id;
    long long risk;
};

struct REGISTRATION
{
    int id;
    long long registration;
    long long register_time[5]; //The timestamp each person receive
                                //The time when the patient register [y
ear(yyyy),month(mm),day(dd),hour(hh),minute(mm)]
    int hospital;               //The nearest hospital
    int regist_halfday;         //The number of halfday the patient goe
s into the queue, i.e. the information went into the system
    int withdraw;
    int withdraw_halfday;
    int reinsert_halfday;
    int risk2halfday;
};

struct TREATMENT
{
    int id;
    long long treatment;
    int treat_halfday;          //The number of halfday the patient get
 treated
};
```

And relational database schema is 4 B+ trees, the structs will be stored in the block and be stored in the trees.

(ii) There are 3 treatments shown in the csv files, each with different priority rules.

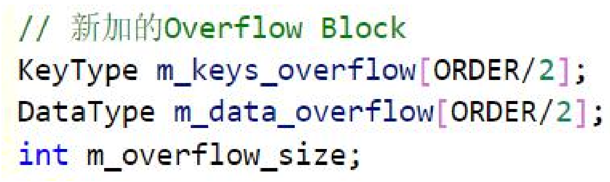
```
void person::calculate_key()
{
    long long age_range;  //Used to store the code of age for key calcu
lation
    age_range = calculate_age_category();
    long long ddl;
    if (ddlletter == 0) { ddl = 2;}
    else {ddl = 1;}
    //This sub-block calculates the key value to used for sort
        // The design of the key is: we combine the indications used fo
r sort into a 10^14 scale number
        // The less the scale is, the higher the priority will be.
        // This first signaficant place is the risk, if the risk is 0,1,
 it is treated as usual, for 2, add one month, for 3, mark
        // it by 2 to leave it till it is empty
        // The second significant place is the profession, 1 for the mo
st important, 8 for the least important
        // The third is the age code, which is declared above
        // The followings are the registration time from year to minute

    if (risk == 0 || risk == 1)
    {
        key = ddl * (long long)pow(10,16)                     //ddl
            + (long long)pow(10, 14)                          //risk code
            + profession * (long long)pow(10, 13)             //profession
            + age_range * (long long)pow(10,12)               //age code
            + register_time[0] * (long long)pow(10,8)         //year
            + register_time[1] * (long long)pow(10,6)         //month
            + register_time[2] * (long long)pow(10,4)         //day
            + register_time[3] * (long long)pow(10,2)         //hour
            + register_time[4];                               //minute
    }
    else if (risk == 2)         //The 1 month delay must be considered
isolatedly, there is a problem, which must be corrected latter on!!!
    {
        key = ddl * (long long)pow(10,16)                      //ddl
            + (long long)pow(10, 14)                           //risk code
            + profession * (long long)pow(10, 13)              //professio
n
            + age_range * (long long)pow(10,12)                //age code
            + register_time[0] * (long long)pow(10,8)          //year
            + register_time[1] * (long long)pow(10,6)      //month
            + register_time[2] * (long long)pow(10,4)          //day
            + register_time[3] * (long long)pow(10,2)          //hour
            + register_time[4];                                //minute
    }
    else if (risk == 3)
    {
        key = ddl * (long long)pow(10,16)                      //ddl
            + 2 * (long long)pow(10, 14)                       //risk code
            + profession * (long long)pow(10,13)               //professio
n
            + age_range * (long long)pow(10,12)                //age code
            + register_time[0] * (long long)pow(10,8)          //year
            + register_time[1] * (long long)pow(10,6)          //month
            + register_time[2] * (long long)pow(10,4)          //day
            + register_time[3] * (long long)pow(10,2)          //hour
            + register_time[4];                                //minute
    }
    key += treatment * (long long)pow(10,15);
}
```

(iii) The Block(main), you can find it in BPlus_node.h and BPlus_node.cpp CNode class



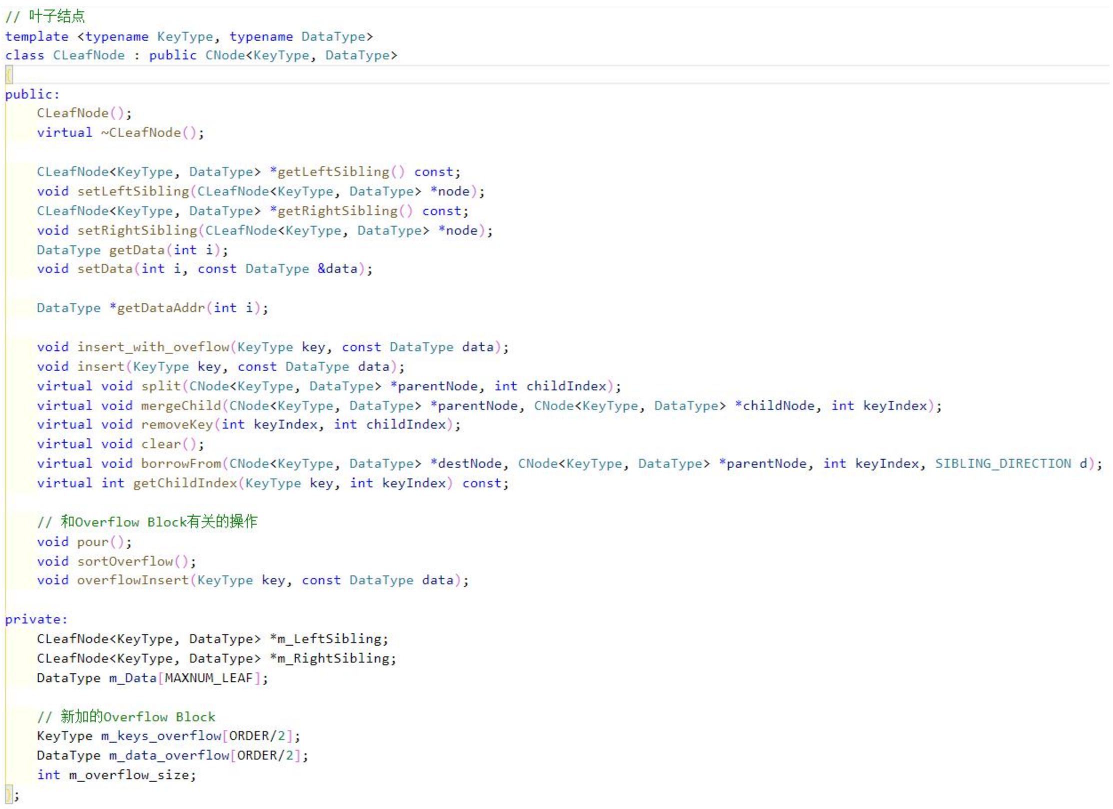
```
protected:
    NODE_TYPE m_Type;
    int m_KeyNum;
    KeyType m_KeyValues[MAXNUM_KEY];
```

(iv) The overflow block can be found in the BPlus_node.h and BPlus_node.cpp CNode class

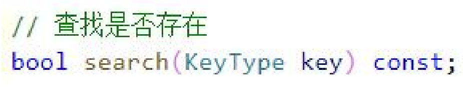


```
// 新加的Overflow Block
KeyType m_keys_overflow[ORDER/2];
DataType m_data_overflow[ORDER/2];
int m_overflow_size;
```

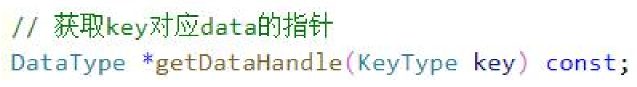
(v) The 6 operations can be found in the BPlus_node.h, BPlus_node.cpp, BPlus_tree.h and BPlus_tree.cpp files.



```
// 叶子结点
template <typename KeyType, typename DataType>
class CLeafNode : public CNode<KeyType, DataType>
{
public:
    CLeafNode();
    virtual ~CLeafNode();

    CLeafNode<KeyType, DataType> *getLeftSibling() const;
    void setLeftSibling(CLeafNode<KeyType, DataType> *node);
    CLeafNode<KeyType, DataType> *getRightSibling() const;
    void setRightSibling(CLeafNode<KeyType, DataType> *node);
    DataType getData(int i);
    void setData(int i, const DataType &data);

    DataType *getDataAddr(int i);

    void insert_with_oveflow(KeyType key, const DataType data);
    void insert(KeyType key, const DataType data);
    virtual void split(CNode<KeyType, DataType> *parentNode, int childIndex);
    virtual void mergeChild(CNode<KeyType, DataType> *parentNode, CNode<KeyType, DataType> *childNode, int keyIndex);
    virtual void removeKey(int keyIndex, int childIndex);
    virtual void clear();
    virtual void borrowFrom(CNode<KeyType, DataType> *destNode, CNode<KeyType, DataType> *parentNode, int keyIndex, SIBLING_DIRECTION d);
    virtual int getChildIndex(KeyType key, int keyIndex) const;

    // 和Overflow Block有关的操作
    void pour();
    void sortOverflow();
    void overflowInsert(KeyType key, const DataType data);

private:
    CLeafNode<KeyType, DataType> *m_LeftSibling;
    CLeafNode<KeyType, DataType> *m_RightSibling;
    DataType m_Data[MAXNUM_LEAF];

    // 新加的Overflow Block
    KeyType m_keys_overflow[ORDER/2];
    DataType m_data_overflow[ORDER/2];
    int m_overflow_size;
};
```



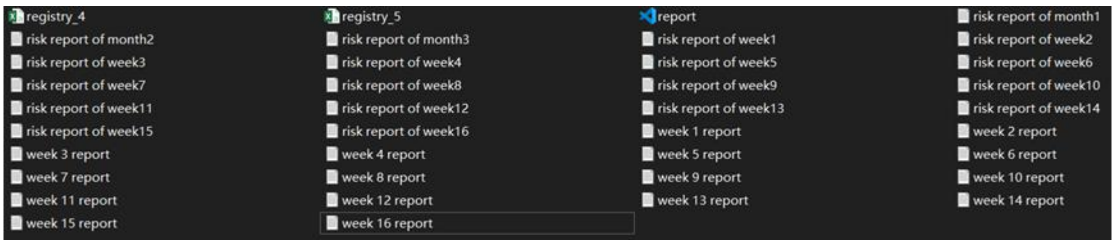
```
// 查找是否存在
bool search(KeyType key) const;
```



```
// 获取key对应data的指针
DataType *getDataHandle(KeyType key) const;
```

EXERCISE 2

We only need to modify the report part. The new various reports are based on the relations and blocks, further, B+/B/hash.



```
registry_4             registry_5             report                 risk report of month1
risk report of month2  risk report of month3  risk report of week1   risk report of week2
risk report of week3   risk report of week4   risk report of week5   risk report of week6
risk report of week7   risk report of week8   risk report of week9   risk report of week10
risk report of week11  risk report of week12  risk report of week13  risk report of week14
risk report of week15  risk report of week16  week 1 report          week 2 report
week 3 report          week 4 report          week 5 report          week 6 report
week 7 report          week 8 report          week 9 report          week 10 report
week 11 report         week 12 report         week 13 report         week 14 report
week 15 report         week 16 report
```

EXERCISE 3

(i) All B+ trees has the same primary key: id.

(ii) We realise them all in B+ tree, shown in main.cpp, BPlus_node.h, BPlus_node.cpp, BPlus_tree.h and BPlus_tree.cpp files.

(iii) Shown in exercise 1

(iv) The B tree uses a new added attribute, health care card as the secondary key and hash table uses risk as its secondary key.

(v) Shown in main.cpp, B_Tree_Decl.h , B_Tree.h , hashset.h , hashset.cpp, BPlus_node.h, BPlus_node.cpp, BPlus_tree.h and BPlus_tree.cpp etc.