CSCE 221 Assignment 5 Cover Page

First Name Peng Last Name Li UIN 822003660

User Name abc E-mail address billlipeng@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People		Lijun Pu(CE Master student)		
Web pages (provide URL)				
Printed material	CSCE221 Slides			
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Peng Li Date Nov 20

Program Description

This lab includes two parts. First part is implement a minimum priority queue with linked-list or unsorted array. I implemented with the linked-list Second part is implement a minimum priority queue with a binary heap.

Purpose of the Assignment

The purpose of this programming assignment is to learn minimum priority queue structure, binary heap structure and the basic operations of a minimum priority queue.

Data Structures Description

We used minimum priority queue in this assignment. During the implementation of the minimum priority queue with linked-list, I used the std::list. Also I used a position class to keep track of the element position in the minimum priority queue. During the implementation of the minimum priority with the binary heap, I used a locator vector to the store the address of each element.

Algorithm Description

1. removeKey()
   Linked-list: Since the linked-list is sorted, removeKey will remove the first element. It costs one operation. O(1).
   Binary Heap: First put the last element at the beginning of the heap O(1), then perform a walk-down O(log n). Thus O(log n).

2. decreaseKey()
   Linked-list: Delete the old element and then insert the new value. The deletion costs O(1), and the insert costs O(n). Thus it is O(n).
   Binary Heap: First decrease the key O(1), then perform a walk-up O(log n). Thus O(log n).

Program Organization and Description of Classes

Linked-list

```
template <typename E>
class PQList { // priority queue list
protected:
    typedef std::list<E> ElementList; // list for the element

public:
```

```
    class Locator{ // Locator
        friend class PQList;
        private:
            typename ElementList::iterator iterator;
        public:
            const E& operator*() { return *iterator; } // content
    };

    Locator insertItem( E& e); // insert element
    void removeKey();
    void printPQList();
    void minKey();
    void createPriorityQueueList(std::string
filename,std::vector<Locator>& r);
    void decreaseKey(const Locator& p, const int& k,
std::vector<Locator>& r);

private:
    ElementList L; // priority queue contents
};

/*****************************************************************
**/
// Item class
class Item {
private:
    int key;
    std::string element;
    PQList<Item>::Locator* p;
public:
    Item(int k, std::string elem):key(k),element(elem){}
    int getKey() const{ return key;}
    std::string getElem() const{return element;}
    PQList<Item>::Locator* getItem(){return p;}
    void setKey(const int& k){key=k;}
    void setElem(std::string& elem){element=elem;}
    void setLocator(PQList<Item>::Locator* pos){p=pos;}
    // less than
    bool operator()(const Item& kep) {return key < kep.key;}
};
```

Heap

```
/*****************************************************************
**/
template <typename E> // insert item
typename PQList<E>::Locator
PQList<E>::insertItem( E& e) {
    typename ElementList::iterator iterator = L.begin();
    while (iterator != L.end() && !e(*iterator))
        ++iterator; // find largest element
    Locator position;
    position.iterator =L.insert(iterator, e); // insert after the
largest
    e.setLocator(&position);
    return position; // inserted position returned to the vector
}


/*****************************************************************
**/
//class Locator
template<typename ElemType>
class Locator{
private:
    Item<ElemType>* item;
public:
    void setItem(Item<ElemType>* i){item=i;}
    Locator(Item<ElemType>* i =NULL): item(i){}
    Item<ElemType>* getItem(){return item;}
};

/*****************************************************************
**/
//class Item
template<typename ElemType>
class Item{
private:
    int key;
    ElemType elem;
    // locator
    Locator<ElemType>* loc;
public:
```

```
    Item( const int k=0,const ElemType& e=ElemType(),
Locator<ElemType>* l=NULL): key(k), elem(e), loc(l) {}
    const int getKey() const { return key; }
    const ElemType& getElem() const { return elem; }
    void setKey( int k) { key = k; }
    void setElem(ElemType& e) { elem = e; }
    // locator
    void setLocator(Locator<ElemType>* l){loc=l;}
    Locator<ElemType>* getLocator(){return loc;}
};

/*****************************************************************
**/
//Class PQComp
template<typename ElemType>
class PQComp {
public:
    int operator()( const Item<ElemType>& e1,const Item<ElemType>&
e2) {
        return e1.getKey() - e2.getKey();
    }
};

/*****************************************************************
**/
//Class Binary Heap
template<typename ElemType, typename Comp, typename L>
class BinaryHeap {
private:
    Comp comp;
    int curSize;
    ElemType *array;
    int length;
    static const int DEF_SIZE = 8;
    void getNewArray(int newSize) {
        array = new ElemType[newSize];
        length = newSize;
    }
public:
    BinaryHeap(int size = DEF_SIZE) { //constructor
        curSize = 0;
```

```
        getNewArray(size);
    }
    ElemType& findMin() { return array[0];}
    bool isEmpty( ) const { return curSize == 0; }
    void createPriorityQueue(string file, Locator<L> locArray[]);
    void insertItem(const ElemType& x,Locator<L>& loc);
    void removeKey();
    void walkUp(int hole, int& count);
    void walkDown(int hole, int& count);
    void decreaseKey( Locator<L>& loc , int k);
    void printPQHeap();
};
```

Instructions to Compile and Run your Program

A makefile is included in the folder. Please follow the commands below to run the program.

$ make
$ ./Main

Input and Output Specifications

The size of the input file is limited to 256 lines.

Logical Exceptions

No logical error has been found in testing.

C++ object oriented or generic programming features

In this lab, object oriented programming features is used. The classes in this lab are shown in Program Organization and Description of Classes section of this lab report.

Analysis

Both implementations results are following the theoretical results. The removeKey for Linked-list take one operation and for binary heap takes log n operations.

Function	Linked-list	Binary Heap
createMPQ	$O(n^2)$	O(n log n)
insertItem	O(n)	O(log n)
minKey	O(1)	O(1)
removeKey	O(1)	O(log n)
decreaseKey	O(n)	O(log n)

Applications

1. CPU scheduler use minimum priority queue for scheduling. Especially the shortest first scheduler.
2. The exam grade can be put into a minimum priority queue for rankings.
3. In hospital, we can assign each patient with a key number according to his or her health condition. Then the patient with higher priority key can see the doctor first.