```
// Lina Kang
// CS1D MW 2:30 - 5:00 PM
// Assignment 6 - Priority Queues
//
// This program simulates an example situation of a queue line
// for the emergency room. This program utilizes the heap
// and a priority queue for its benefits with efficiency
// since only the element with the highest priority is necessary
// rather than the rest of the elements and their specific order

/* OUTPUT
 -----------------------------------------
 -- Developed Priority Queue simulation --
 -----------------------------------------

Enter Patient's Name (Type -1 to quit): Bob Bleeding
Enter Patient's Waiting Time: 2
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Frank Feelingbad
Enter Patient's Waiting Time: 3
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Cathy Coughing
Enter Patient's Waiting Time: 5
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Sam Sneezing
Enter Patient's Waiting Time: 10
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 1:12

Enter Patient's Name (Type -1 to quit): Paula Pain
Enter Patient's Waiting Time: 10
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 2:19

Enter Patient's Name (Type -1 to quit): Sid Sickly
Enter Patient's Waiting Time: 4
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Alice Ailment
Enter Patient's Waiting Time: 7
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Irene Ill
Enter Patient's Waiting Time: 1
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Tom Temperature
Enter Patient's Waiting Time: 6
```

```
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): -1
--------------------------------
Current Emergency Room:

Name              | Waiting Time
---------------------------------
Alice Ailment     | 7
Cathy Coughing    | 5
Tom Temperature   | 6
Bob Bleeding      | 2
Sid Sickly        | 4
Irene Ill         | 1
Frank Feelingbad  | 3
--------------------------------

The Afternoon Has Begun.

--------------------------------
Current Patient: Alice Ailment
Start Time: 12:00
End Time:   12:25
--------------------------------
Current Patient: Tom Temperature
Start Time: 12:25
End Time:   12:50
--------------------------------
Current Patient: Cathy Coughing
Start Time: 12:50

*********************************
** Life Threatening!!
** Current Patient: Sam Sneezing
** Start Time: 1:12
** End Time: 1:37
*********************************

End Time:   1:40
--------------------------------
Current Patient: Sid Sickly
Start Time: 1:40
End Time:   2:05
--------------------------------
Current Patient: Frank Feelingbad
Start Time: 2:05

*********************************
** Life Threatening!!
** Current Patient: Paula Pain
** Start Time: 2:19
** End Time: 2:44
*********************************

End Time:   2:55
--------------------------------
```

```
Current Patient: Bob Bleeding
Start Time: 2:55
End Time:   3:20
--------------------------------
Current Patient: Irene Ill
Start Time: 3:20
End Time:   3:45


 -----------------------------------------
 --    STL priority queue simulation    --
 -----------------------------------------

Enter Patient's Name (Type -1 to quit): Bob Bleeding
Enter Patient's Waiting Time: 2
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Frank Feelingbad
Enter Patient's Waiting Time: 3
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Cathy Coughing
Enter Patient's Waiting Time: 5
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Sam Sneezing
Enter Patient's Waiting Time: 10
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 1:12

Enter Patient's Name (Type -1 to quit): Paula Pain
Enter Patient's Waiting Time: 10
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 2:19

Enter Patient's Name (Type -1 to quit): Sid Sickly
Enter Patient's Waiting Time: 4
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Alice Ailment
Enter Patient's Waiting Time: 7
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Irene Ill
Enter Patient's Waiting Time: 1
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000

Enter Patient's Name (Type -1 to quit): Tom Temperature
Enter Patient's Waiting Time: 6
If the patient is life-threatening, enter time.
If not, type 000 and press enter: 000
```

```

Enter Patient's Name (Type -1 to quit): -1
--------------------------------
Current Emergency Room:

Name                  | Waiting Time
---------------------------------
Alice Ailment         | 7
Tom Temperature       | 6
Cathy Coughing        | 5
Sid Sickly            | 4
Frank Feelingbad      | 3
Bob Bleeding          | 2
Irene Ill             | 1

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

The Afternoon Has Begun.

--------------------------------
Current Patient: Alice Ailment
Start Time: 12:00
End Time:   12:25
--------------------------------
Current Patient: Tom Temperature
Start Time: 12:25
End Time:   12:50
--------------------------------
Current Patient: Cathy Coughing
Start Time: 12:50

*********************************
** Life Threatening!!
** Current Patient: Sam Sneezing
** Start Time: 1:12
** End Time: 1:37
*********************************

End Time:   1:40
--------------------------------
Current Patient: Sid Sickly
Start Time: 1:40
End Time:   2:05
--------------------------------
Current Patient: Frank Feelingbad
Start Time: 2:05

*********************************
** Life Threatening!!
** Current Patient: Paula Pain
** Start Time: 2:19
** End Time: 2:44
*********************************

End Time:   2:55
--------------------------------
Current Patient: Bob Bleeding
```

```cpp
Start Time: 2:55
End Time:   3:20
--------------------------------
Current Patient: Irene Ill
Start Time: 3:20
End Time:   3:45

 */

#include "header.h"

int main()
{
    cout << "\n -----------------------------------------"
            "\n -- Developed Priority Queue simulation --"
            "\n -----------------------------------------\n\n";

    // Make priority queue / vectors and Input Patients

    priorityQueue emergencyQueue;
    vector<Patient> lifeThreateningPatients;

    Patient * temp;
    string name;
    string lifeThreat;
    int waitTime;

    cout << "Enter Patient's Name (Type -1 to quit): ";
    getline(cin, name);

    while(name != "-1")
    {
        cout << "Enter Patient's Waiting Time: ";
        cin >> waitTime;
        cin.ignore(1000, '\n');
        cout << "If the patient is life-threatening, enter time.\n"
                "If not, type 000 and press enter: ";
        getline(cin, lifeThreat);

        temp = new Patient {name, waitTime, lifeThreat};

        if(lifeThreat == "000")
        {
            emergencyQueue.push(*temp);
        }
        else
        {
            lifeThreateningPatients.push_back(*temp);
        }
        cout << endl;
        cout << "Enter Patient's Name (Type -1 to quit): ";
        getline(cin, name);
    }

    // Set up clock and timer

    Time clockTime;
    clockTime.set(12, 0);
    int timerTime = 0;          // 25 minute limit timer for a single patient

    // Print the current priority queue

    cout << "--------------------------------\n"
            "Current Emergency Room: \n";

    emergencyQueue.print();

    cout << "--------------------------------\n"
            "\nThe Afternoon Has Begun.\n\n";

    // Begin treating patients

    while(emergencyQueue.getSize() > 0)
    {
        Patient current = emergencyQueue.top();
        cout << "--------------------------------\n"
                "Current Patient: " << current.name << endl;
        cout << "Start Time: " << clockTime.get() << endl;

        // Working with the patient...
        while(timerTime < 25)       // Pass 25 minutes of time for treatment
        {
            timerTime++;
            clockTime.increment();

            // Check if life-threatening situation will happen at this time
            checkLifeThreatening(emergencyQueue, clockTime, lifeThreateningPatients);
        }

        cout << "End Time:   " << clockTime.get() << endl;
        timerTime = 0;
        emergencyQueue.pop();
    }

    cout << "\n"
            "\n -----------------------------------------"
            "\n --    STL priority queue simulation    --"
            "\n -----------------------------------------\n\n";

    priority_queue<Patient, vector<Patient>, ComparePatient> STL_queue;

    cout << "Enter Patient's Name (Type -1 to quit): ";
    getline(cin, name);

    while(name != "-1")
    {
        cout << "Enter Patient's Waiting Time: ";
        cin >> waitTime;
        cin.ignore(1000, '\n');
        cout << "If the patient is life-threatening, enter time.\n"
                "If not, type 000 and press enter: ";
        getline(cin, lifeThreat);

        temp = new Patient {name, waitTime, lifeThreat};

        if(lifeThreat == "000")
        {
            STL_queue.push(*temp);
        }
        else
        {
            lifeThreateningPatients.push_back(*temp);
        }
        cout << endl;
        cout << "Enter Patient's Name (Type -1 to quit): ";
        getline(cin, name);
    }

    clockTime.set(12, 0);
    timerTime = 0;

    cout << "--------------------------------\n"
            "Current Emergency Room: \n";

    print_queue(STL_queue);

    cout << "--------------------------------\n"
            "\nThe Afternoon Has Begun.\n\n";

    while(STL_queue.size() > 0)
    {
        Patient current = STL_queue.top();
        cout << "--------------------------------\n"
                "Current Patient: " << current.name << endl;
        cout << "Start Time: " << clockTime.get() << endl;

        while(timerTime < 25)
        {
            timerTime++;
            clockTime.increment();
            checkLifeThreatening(STL_queue, clockTime, lifeThreateningPatients);
        }

        cout << "End Time:   " << clockTime.get() << endl;
        timerTime = 0;
        STL_queue.pop();
    }

}


#ifndef HEADER_H_
#define HEADER_H_

#include <iostream>
#include <vector>
#include <iomanip>
#include <queue>

using namespace std;

```

```
//--------------- Patient Object ---------------//

struct Patient
{
    string name;
    int waitingTime;
    string lifeThreatening;

    // comparators to define a priority of a patient in the queue
    bool operator<(Patient compared)
    {
        if(waitingTime < compared.waitingTime)
            return true;
        else
            return false;
    }
    bool operator>(Patient compared)
    {
        if(waitingTime > compared.waitingTime)
            return true;
        else
            return false;
    }
};

//--------------- A way to define Time ---------------//

class Time
{
private:
    int hour;
    int minTen;
    int minOne;
public:
    Time();
    string get();
    void set(int hour, int min);
    void increment();
};
Time::Time()
{
    hour = 12;
    minTen = 0;
    minOne = 0;
}
string Time::get()
{
    return to_string(hour) + ":" + to_string(minTen) + to_string(minOne);
}
void Time::set(int hourInput, int min)
{
    hour = hourInput;
    if(min < 10)
    {
        minTen = 0;
        minOne = min;
    }
    else
    {
        minTen = min/10;
        minOne = min%10;
    }
}
void Time::increment()
{
    minOne++;

    if(minOne == 10)
    {
        minOne = 0;
        minTen++;
        if(minTen == 6)
        {
            minTen = 0;
            hour++;
            if(hour == 13)
                hour = 1;
        }
    }
}

//--------------- For STL priority_queue --------------//

//inform STL priority_queue how to compare Patient objects
class ComparePatient
{
public:
    bool operator()(Patient & p1, Patient & p2)
    {
       if (p1.waitingTime < p2.waitingTime)
           return true;
       return false;
    }
};
// print function for STL priority_queue
template <class T>
void print_queue(T q)
{
    cout << endl << left << setw(17) <<"Name" << " | " << "Waiting Time\n"
            "---------------------------------\n";
    while(!q.empty())
    {
        cout << setw(17) << q.top().name << " | " << q.top().waitingTime << endl;
        q.pop();
    }
    cout << '\n';
}

//--------------- Developed Priority Queue --------------//

class priorityQueue
{
public:
    priorityQueue();
    void push(Patient);
    void pop();
    Patient top();
    int getSize();
    bool isEmpty();

    void swap(int, int);

    void heapUp(int index);
    void heapDown(int index);

    void print();

private:
    vector<Patient> vect;
    Patient emptyFirst;
    int size;

};

priorityQueue::priorityQueue()
{
    // The index 0 will remain an empty
    // From index 1, patients will be added in
    emptyFirst.name = "NOBODY";
    emptyFirst.waitingTime = 100;
    emptyFirst.lifeThreatening = "";

    vect.push_back(emptyFirst);
    size = 0;
}
void priorityQueue::push(Patient newPatient)
{
    vect.push_back(newPatient);
    heapUp(vect.size()-1);
    size++;
}
void priorityQueue::pop()
{
    swap(1, size);
    vect.pop_back();
    heapDown(1);
    size--;
}
Patient priorityQueue::top()
{
    return vect[1];
}
int priorityQueue::getSize()
{
    return size;
}
bool priorityQueue::isEmpty()
{
    return size == 0;
}

```

```cpp
void priorityQueue::heapUp(int index)
{
    int parentIndex = index/2;
    if(vect[parentIndex] < vect[index])
    {
        swap(parentIndex, index);
        heapUp(parentIndex);
    }
}
void priorityQueue::heapDown(int index)
{
    int left  = 2 * index;
    int right = 2 * index + 1;
    int largest = index;

    if(left < size && vect[left] > vect[index] )
        largest = left;
    if(right < size && vect[right] > vect[largest])
        largest = right;

    if(largest != index)
    {
        swap(index, largest);
        heapDown(largest);
    }
}

void priorityQueue::swap(int parent, int child)
{
    Patient temp = vect[parent];
    vect[parent] = vect[child];
    vect[child] = temp;
}
void priorityQueue::print()
{
    cout << endl << left << setw(17) <<"Name" << " | " << "Waiting Time\n"
            "---------------------------------\n";
    for(int i = 1; i <= size; i++)
    {
        cout << setw(17) << vect[i].name << " | " << vect[i].waitingTime << endl;
    }
}

//-------------- Additional Helper Functions --------------//

// At every passing minute, checks whether it is time for life-threatening
// interruption. If so, life-threatening patients will be treated at
// that moment and delay the rest of the patients 25 minutes.
template <class priorityQueueTemplate>
void checkLifeThreatening(priorityQueueTemplate & emergencyQueue,
        Time & clockTime, vector<Patient> & lifeThreateningPatients)
{
    // traverse through array of life-threatening patients
    for(int i= 0; i < lifeThreateningPatients.size(); i++)
    {
        // if the current time matches the patient's interruption time..
        if (clockTime.get() == lifeThreateningPatients[i].lifeThreatening)
        {
            cout << "\n*********************************"
                    "\n** Life Threatening!!";

            // push the patient onto the general priority queue
            // (will be put to the top, as highest priority)
            emergencyQueue.push(lifeThreateningPatients[i]);

            cout << "\n** Current Patient: " << emergencyQueue.top().name;
            cout << "\n** Start Time: " << clockTime.get();

            // delay 25 minutes
            for(int i = 0; i < 25; i++)
                clockTime.increment();

            // remove patient from both lists
            emergencyQueue.pop();
            lifeThreateningPatients.erase(lifeThreateningPatients.begin());

            cout << "\n** End Time: " << clockTime.get() <<
                    "\n*********************************\n\n";
        }
    }
}

#endif /* HEADER_H_ */


```