```
// 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
--------------------------------
```

```
115 Current Patient: Bob Bleeding
116 Start Time: 2:55
117 End Time:   3:20
118 --------------------------------
119 Current Patient: Irene Ill
120 Start Time: 3:20
121 End Time:   3:45
122
123
124  ----------------------------------------
125  --    STL priority queue simulation    --
126  ----------------------------------------
127
128 Enter Patient's Name (Type -1 to quit): Bob Bleeding
129 Enter Patient's Waiting Time: 2
130 If the patient is life-threatening, enter time.
131 If not, type 000 and press enter: 000
132
133 Enter Patient's Name (Type -1 to quit): Frank Feelingbad
134 Enter Patient's Waiting Time: 3
135 If the patient is life-threatening, enter time.
136 If not, type 000 and press enter: 000
137
138 Enter Patient's Name (Type -1 to quit): Cathy Coughing
139 Enter Patient's Waiting Time: 5
140 If the patient is life-threatening, enter time.
141 If not, type 000 and press enter: 000
142
143 Enter Patient's Name (Type -1 to quit): Sam Sneezing
144 Enter Patient's Waiting Time: 10
145 If the patient is life-threatening, enter time.
146 If not, type 000 and press enter: 1:12
147
148 Enter Patient's Name (Type -1 to quit): Paula Pain
149 Enter Patient's Waiting Time: 10
150 If the patient is life-threatening, enter time.
151 If not, type 000 and press enter: 2:19
152
153 Enter Patient's Name (Type -1 to quit): Sid Sickly
154 Enter Patient's Waiting Time: 4
155 If the patient is life-threatening, enter time.
156 If not, type 000 and press enter: 000
157
158 Enter Patient's Name (Type -1 to quit): Alice Ailment
159 Enter Patient's Waiting Time: 7
160 If the patient is life-threatening, enter time.
161 If not, type 000 and press enter: 000
162
163 Enter Patient's Name (Type -1 to quit): Irene Ill
164 Enter Patient's Waiting Time: 1
165 If the patient is life-threatening, enter time.
166 If not, type 000 and press enter: 000
167
168 Enter Patient's Name (Type -1 to quit): Tom Temperature
169 Enter Patient's Waiting Time: 6
170 If the patient is life-threatening, enter time.
171 If not, type 000 and press enter: 000
```

```
172
173 Enter Patient's Name (Type -1 to quit): -1
174 --------------------------------
175 Current Emergency Room:
176
177 Name              | Waiting Time
178 ---------------------------------
179 Alice Ailment     | 7
180 Tom Temperature   | 6
181 Cathy Coughing    | 5
182 Sid Sickly        | 4
183 Frank Feelingbad  | 3
184 Bob Bleeding      | 2
185 Irene Ill         | 1
186
187 --------------------------------
188
189 The Afternoon Has Begun.
190
191 --------------------------------
192 Current Patient: Alice Ailment
193 Start Time: 12:00
194 End Time:   12:25
195 --------------------------------
196 Current Patient: Tom Temperature
197 Start Time: 12:25
198 End Time:   12:50
199 --------------------------------
200 Current Patient: Cathy Coughing
201 Start Time: 12:50
202
203 ********************************
204 ** Life Threatening!!
205 ** Current Patient: Sam Sneezing
206 ** Start Time: 1:12
207 ** End Time: 1:37
208 ********************************
209
210 End Time:   1:40
211 --------------------------------
212 Current Patient: Sid Sickly
213 Start Time: 1:40
214 End Time:   2:05
215 --------------------------------
216 Current Patient: Frank Feelingbad
217 Start Time: 2:05
218
219 ********************************
220 ** Life Threatening!!
221 ** Current Patient: Paula Pain
222 ** Start Time: 2:19
223 ** End Time: 2:44
224 ********************************
225
226 End Time:   2:55
227 --------------------------------
228 Current Patient: Bob Bleeding
```

```
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;
```

```
286     clockTime.set(12, 0);
287     int timerTime = 0;          // 25 minute limit timer for a single patient
288
289     // Print the current priority queue
290
291     cout << "--------------------------------\n"
292             "Current Emergency Room: \n";
293
294     emergencyQueue.print();
295
296     cout << "--------------------------------\n"
297             "\nThe Afternoon Has Begun.\n\n";
298
299     // Begin treating patients
300
301     while(emergencyQueue.getSize() > 0)
302     {
303         Patient current = emergencyQueue.top();
304         cout << "--------------------------------\n"
305                 "Current Patient: " << current.name << endl;
306         cout << "Start Time: " << clockTime.get() << endl;
307
308         // Working with the patient...
309         while(timerTime < 25)       // Pass 25 minutes of time for treatment
310         {
311             timerTime++;
312             clockTime.increment();
313
314             // Check if life-threatening situation will happen at this time
315             checkLifeThreatening(emergencyQueue, clockTime, lifeThreateningPatients);
316         }
317
318         cout << "End Time:   " << clockTime.get() << endl;
319         timerTime = 0;
320         emergencyQueue.pop();
321     }
322
323     cout << "\n"
324             "\n ----------------------------------------"
325             "\n --    STL priority queue simulation    --"
326             "\n ----------------------------------------\n\n";
327
328     priority_queue<Patient, vector<Patient>, ComparePatient> STL_queue;
329
330     cout << "Enter Patient's Name (Type -1 to quit): ";
331     getline(cin, name);
332
333     while(name != "-1")
334     {
335         cout << "Enter Patient's Waiting Time: ";
336         cin >> waitTime;
337         cin.ignore(1000, '\n');
338         cout << "If the patient is life-threatening, enter time.\n"
339                 "If not, type 000 and press enter: ";
340         getline(cin, lifeThreat);
341
342         temp = new Patient {name, waitTime, lifeThreat};
```

```
343
344         if(lifeThreat == "000")
345         {
346             STL_queue.push(*temp);
347         }
348         else
349         {
350             lifeThreateningPatients.push_back(*temp);
351         }
352         cout << endl;
353         cout << "Enter Patient's Name (Type -1 to quit): ";
354         getline(cin, name);
355     }
356
357     clockTime.set(12, 0);
358     timerTime = 0;
359
360     cout << "--------------------------------\n"
361             "Current Emergency Room: \n";
362
363     print_queue(STL_queue);
364
365     cout << "--------------------------------\n"
366             "\nThe Afternoon Has Begun.\n\n";
367
368     while(STL_queue.size() > 0)
369     {
370         Patient current = STL_queue.top();
371         cout << "--------------------------------\n"
372                 "Current Patient: " << current.name << endl;
373         cout << "Start Time: " << clockTime.get() << endl;
374
375         while(timerTime < 25)
376         {
377             timerTime++;
378             clockTime.increment();
379             checkLifeThreatening(STL_queue, clockTime, lifeThreateningPatients);
380         }
381
382         cout << "End Time:   " << clockTime.get() << endl;
383         timerTime = 0;
384         STL_queue.pop();
385     }
386
387 }
388
389
390 #ifndef HEADER_H_
391 #define HEADER_H_
392
393 #include <iostream>
394 #include <vector>
395 #include <iomanip>
396 #include <queue>
397
398 using namespace std;
399
```

```
400 //--------------- Patient Object ---------------//
401
402 struct Patient
403 {
404     string name;
405     int waitingTime;
406     string lifeThreatening;
407
408     // comparators to define a priority of a patient in the queue
409     bool operator<(Patient compared)
410     {
411         if(waitingTime < compared.waitingTime)
412             return true;
413         else
414             return false;
415     }
416     bool operator>(Patient compared)
417     {
418         if(waitingTime > compared.waitingTime)
419             return true;
420         else
421             return false;
422     }
423 };
424
425 //--------------- A way to define Time ---------------//
426
427 class Time
428 {
429 private:
430     int hour;
431     int minTen;
432     int minOne;
433 public:
434     Time();
435     string get();
436     void set(int hour, int min);
437     void increment();
438 };
439 Time::Time()
440 {
441     hour = 12;
442     minTen = 0;
443     minOne = 0;
444 }
445 string Time::get()
446 {
447     return to_string(hour) + ":" + to_string(minTen) + to_string(minOne);
448 }
449 void Time::set(int hourInput, int min)
450 {
451     hour = hourInput;
452     if(min < 10)
453     {
454         minTen = 0;
455         minOne = min;
456     }
```

```
457     else
458     {
459         minTen = min/10;
460         minOne = min%10;
461     }
462 }
463 void Time::increment()
464 {
465     minOne++;
466 
467     if(minOne == 10)
468     {
469         minOne = 0;
470         minTen++;
471         if(minTen == 6)
472         {
473             minTen = 0;
474             hour++;
475             if(hour == 13)
476                 hour = 1;
477         }
478     }
479 }
480 
481 //--------------- For STL priority_queue --------------//
482 
483 //inform STL priority_queue how to compare Patient objects
484 class ComparePatient
485 {
486 public:
487     bool operator()(Patient & p1, Patient & p2)
488     {
489        if (p1.waitingTime < p2.waitingTime)
490            return true;
491        return false;
492     }
493 };
494 // print function for STL priority_queue
495 template <class T>
496 void print_queue(T q)
497 {
498     cout << endl << left << setw(17) <<"Name" << " | " << "Waiting Time\n"
499             "--------------------------------\n";
500     while(!q.empty())
501     {
502         cout << setw(17) << q.top().name << " | " << q.top().waitingTime << endl;
503         q.pop();
504     }
505     cout << '\n';
506 }
507 
508 //--------------- Developed Priority Queue --------------//
509 
510 class priorityQueue
511 {
512 public:
513     priorityQueue();
```

```
514     void push(Patient);
515     void pop();
516     Patient top();
517     int getSize();
518     bool isEmpty();
519 
520     void swap(int, int);
521 
522     void heapUp(int index);
523     void heapDown(int index);
524 
525     void print();
526 
527 private:
528     vector<Patient> vect;
529     Patient emptyFirst;
530     int size;
531 
532 };
533 
534 priorityQueue::priorityQueue()
535 {
536     // The index 0 will remain an empty
537     // From index 1, patients will be added in
538     emptyFirst.name = "NOBODY";
539     emptyFirst.waitingTime = 100;
540     emptyFirst.lifeThreatening = "";
541 
542     vect.push_back(emptyFirst);
543     size = 0;
544 }
545 void priorityQueue::push(Patient newPatient)
546 {
547     vect.push_back(newPatient);
548     heapUp(vect.size()-1);
549     size++;
550 }
551 void priorityQueue::pop()
552 {
553     swap(1, size);
554     vect.pop_back();
555     heapDown(1);
556     size--;
557 }
558 Patient priorityQueue::top()
559 {
560     return vect[1];
561 }
562 int priorityQueue::getSize()
563 {
564     return size;
565 }
566 bool priorityQueue::isEmpty()
567 {
568     return size == 0;
569 }
570 
```

```
571 void priorityQueue::heapUp(int index)
572 {
573     int parentIndex = index/2;
574     if(vect[parentIndex] < vect[index])
575     {
576         swap(parentIndex, index);
577         heapUp(parentIndex);
578     }
579 }
580 void priorityQueue::heapDown(int index)
581 {
582     int left  = 2 * index;
583     int right = 2 * index + 1;
584     int largest = index;
585 
586     if(left < size && vect[left] > vect[index] )
587         largest = left;
588     if(right < size && vect[right] > vect[largest])
589         largest = right;
590 
591     if(largest != index)
592     {
593         swap(index, largest);
594         heapDown(largest);
595     }
596 }
597 
598 void priorityQueue::swap(int parent, int child)
599 {
600     Patient temp = vect[parent];
601     vect[parent] = vect[child];
602     vect[child] = temp;
603 }
604 void priorityQueue::print()
605 {
606     cout << endl << left << setw(17) <<"Name" << " | " << "Waiting Time\n"
607             "---------------------------------\n";
608     for(int i = 1; i <= size; i++)
609     {
610         cout << setw(17) << vect[i].name << " | " << vect[i].waitingTime << endl;
611     }
612 }
613 
614 //-------------- Additional Helper Functions --------------//
615 
616 // At every passing minute, checks whether it is time for life-threatening
617 // interruption. If so, life-threatening patients will be treated at
618 // that moment and delay the rest of the patients 25 minutes.
619 template <class priorityQueueTemplate>
620 void checkLifeThreatening(priorityQueueTemplate & emergencyQueue,
621         Time & clockTime, vector<Patient> & lifeThreateningPatients)
622 {
623     // traverse through array of life-threatening patients
624     for(int i= 0; i < lifeThreateningPatients.size(); i++)
625     {
626         // if the current time matches the patient's interruption time..
627         if (clockTime.get() == lifeThreateningPatients[i].lifeThreatening)
```

```
628         {
629             cout << "\n*********************************"
630                     "\n** Life Threatening!!";
631 
632             // push the patient onto the general priority queue
633             // (will be put to the top, as highest priority)
634             emergencyQueue.push(lifeThreateningPatients[i]);
635 
636             cout << "\n** Current Patient: " << emergencyQueue.top().name;
637             cout << "\n** Start Time: " << clockTime.get();
638 
639             // delay 25 minutes
640             for(int i = 0; i < 25; i++)
641                 clockTime.increment();
642 
643             // remove patient from both lists
644             emergencyQueue.pop();
645             lifeThreateningPatients.erase(lifeThreateningPatients.begin());
646 
647             cout << "\n** End Time: " << clockTime.get() <<
648                     "\n*********************************\n\n";
649         }
650     }
651 }
652 
653 #endif /* HEADER_H_ */
654 
655 
```