The functionalities an SL\_PriorityQueue should provide are that you should be able to construct and destruct any objects, enter items into the queue and then remove items from the queue and lastly it should be able to display the value contained at the front of the queue without removing it.

|  |
| --- |
| SL\_PriorityQueue: is a LinkedSortedList  Template: dataType |
| Constructor : Object  Copy Constructor : Object  Destructor : void  isEmpty() : bool  enqueue() : bool  dequeue() : bool  peekFront() : dataType  showQueue() : void |

The stubs of the functions are as follows.

template<typename dataType>

SL\_PriorityQueue<dataType>::SL\_PriorityQueue() : sListPtr(new LinkedSortedList<dataType>()){

}

template<typename dataType>

SL\_PriorityQueue<dataType>::SL\_PriorityQueue(const SL\_PriorityQueue& pq){

}

template<typename dataType>

SL\_PriorityQueue<dataType>::~SL\_PriorityQueue(){

}

template<typename dataType>

bool SL\_PriorityQueue<dataType>::isEmpty() const{

}

template<typename dataType>

bool SL\_PriorityQueue<dataType>::enqueue(const dataType& newEntry){

}

template<typename dataType>

bool SL\_PriorityQueue<dataType>::dequeue(){

}

template<typename dataType>

dataType SL\_PriorityQueue<dataType>::peekFront() const {

}

template<typename dataType>

void SL\_PriorityQueue<dataType>::showQueue() const{

}