```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   SLinkedListSE.h
 * Author: LTSACH
 *
 * Created on 31 August 2020, 21:00
 */

#ifndef SLINKEDLISTSE_H
#define SLINKEDLISTSE_H

#include "list/SLinkedList.h"
#include "sorting/ISort.h"

template <class T>
class SLinkedListSE : public SLinkedList<T>
{
public:
    SLinkedListSE(
        void (*removeData)(SLinkedList<T> *) = 0,
        bool (*itemEQ)(T &, T &) = 0) : SLinkedList<T>(removeData, itemEQ){

```

```
28                                      };
29      void sort(int (*comparator)(T &, T &) = 0)
30      {
31          if (this->count > 0)
32          {
33              typename SLinkedList<T>::Node *first = this->head->next; //first user's dat
34              this->tail->next->next = 0;                              //to tail => to 0
35              mergeSort(first, comparator);
36
37              //find the last user's data
38              typename SLinkedList<T>::Node *last = first;
39              while (last->next != 0)
40                  last = last->next;
41
42              //attach to head&tail of the list
43              last->next = this->tail;
44              this->tail->next = last;
45              this->head->next = first;
46          }
47      };
48
49  protected:
50      static int compare(T &lhs, T &rhs, int (*comparator)(T &, T &) = 0)
51      {
52          if (comparator != 0)
53              return comparator(lhs, rhs);
54          else
55          {
56              if (lhs < rhs)
57                  return -1;
58              else if (lhs > rhs)
59                  return +1;
60              else
61                  return 0;
62          }
63      }
64      void mergeSort(typename SLinkedList<T>::Node *&head, int (*comparator)(T &, T &) =
65      {
66          if ((head != 0) && (head->next != 0))
67          {
68              typename SLinkedList<T>::Node *second;
69              devide(head, second);
70              mergeSort(head, comparator);
71              mergeSort(second, comparator);
72              merge(head, second, comparator);
73          }
74      };
75      void devide(typename SLinkedList<T>::Node *&first, typename SLinkedList<T>::Node *&
76      {
77          //YOUR CODE HERE
78          typename SLinkedList<T>::Node *slow, *fast;
79          if (first == NULL)
```

```
80          {
81              return;
82          }
83          else
84          {
85              slow = first;
86              fast = slow->next;
87              while (fast != NULL)
88              {
89                  fast = fast->next;
90                  if(fast!=NULL)
91                  {
92                      slow = slow->next;
93                      fast = fast->next;
94                  }
95              }
96          }
97          second = slow->next;
98          slow->next = NULL;
99      }
100     void merge(typename SLinkedList<T>::Node *&first, typename SLinkedList<T>::Node *&s
101     {
102         if(first == NULL || second == NULL)
103             return;
104         typename SLinkedList<T>::Node *temp = NULL;
105         if(compare(first->data,second->data,comparator) == -1)
106         {
107             temp = first;
108             first = first->next;
109         }
110         else
111         {
112             temp = second;
113             second = second->next;
114         }
115         typename SLinkedList<T>::Node *ptr = temp;
116         while(first!=NULL && second!=NULL)
117         {
118             if(compare(first->data, second->data, comparator) == -1)
119             {
120                 ptr->next = first;
121                 first = first->next;
122             }
123             else
124             {
125                 ptr->next = second;
126                 second = second->next;
127             }
128             ptr = ptr->next;
129         }
130         if(first == NULL && second!=NULL)
131         {
```

```
            ptr->next = second;
        }
        if(second == NULL && first!=NULL)
        {
            ptr->next = first;
        }
        first = temp;
    }
};

#endif /* SLINKEDLISTSE_H */
```