```c
/*
 ================================================================================
 Aufgabe     : Sortieren - Woche 9
 Autor       : Erik Kaufmann
 Matrikel    : 1390365
 Version     : 1.0
 ================================================================================
 */
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include "dhbwsortheap.h"

 //Ab hier Aufgaben

 //Auf true setzen, damit Heapsort getestet wird
bool HeapSortImplemented()
{
    return true;
}

//Hilfsfunktionen
void Swap1(Student_p* studA, Student_p* studB, int indexA, int indexB)
{
    //printf("\nTausche [%d] %s %d mit [%d]%s %d\n", indexB, (*studB)->lastname, (*studB)->matrnr, indexA, (*studA)->lastname, (*studA)->matrnr);
    Student_p* temp = *studA;
    *studA = *studB;
    *studB = temp;
}


//Lasse kleinen Knoten nach unten sinken
void HeapBubbleDown(Student_p* array, int nodeIndex, int end)
{
    // Initialize max as root
    int max = end;

    int leftChild = 2 * end + 1;
    int rightChild = 2 * end + 2;

    // if leftChild is bigger than parent
    if (leftChild < nodeIndex && array[leftChild]->matrnr > array[max]->matrnr)
    {
        max = leftChild;
    }
```

```
46
47      // If rightChild is bigger than parent
48      if (rightChild < nodeIndex && array[rightChild]->matrnr > array
          [max]->matrnr)
49      {
50          max = rightChild;
51      }
52
53      // If max is not root
54      if (max != end)
55      {
56          // swap last mit max
57          Swap1(&array[end], &array[max], end, max);
58
59          // Rekursiver Aufruf, um nach dem Vertauschen weiter zu
              prüfen.
60          HeapBubbleDown(array, nodeIndex, max);
61      }
62
63      return;
64  }
65
66
67  //Stellt Heap-Eigenschaft in einem Array (als Binaerbaum
      interpretiert) her
68  void Heapify(Student_p* array, int count)
69  {
70      int offset = 1;
71      // Auf MaxHeap bringen (Kein Parent ist kleiner als eines seiner
          childs)
72      // Durchlaufe alle 3er Gruppen (Parent + LeftChild + RightChild)
          von unten nach oben und schiebe den größeren Knoten nach
          oben,
73      // bzw. tausche den Parent mit dem größten Child
74      for (int i = count / 2 - offset; i >= 0; i--)
75      {
76          HeapBubbleDown(array, count, i);
77
78          //printf("\n------------------------\n");
79
80          //for (int f = 0; f < count; f++)
81          //{
82          //  printf("[%d] %s %d\n", f, array[f]->lastname, array[f]-
                >matrnr);
83          //}
84
85          //printf("\n------------------------\n");
86      }
87
88      // Heap sort
89      for (int i = count - offset; i > 0; i--) {
90
91          // Tausche Root mit dem letzten element
```

```
92          Swap1(&array[0], &array[i], 0, i);
93
94          // Heapify ab der Wurzel
95          HeapBubbleDown(array, i, 0);
96
97          //printf("\n------------------------\n");
98          //for (int f = 0; f < count; f++)
99          //{
100         //  printf("[%d] %s %d\n", f, array[f]->lastname, array[f]-
              >matrnr);
101         //}
102         //printf("\n------------------------\n");
103     }
104 }
105
106
107 //Heapsort mit Array
108 //Tipp: Hilfsfunktionen benutzen
109 void HeapSortArray(Student_p* array, int count)
110 {
111     //for (int f = 0; f < count; f++)
112     //{
113     //  printf("[%d] %s %d\n", f, array[f]->lastname, array[f]-
          >matrnr);
114     //}
115
116     Heapify(array, count);
117
118     //printf("\n------------------------\n");
119
120     //for (int f = 0; f < count; f++)
121     //{
122     //  printf("[%d] %s %d\n", f, array[f]->lastname, array[f]-
          >matrnr);
123     //}
124 }
125
```