





cs330su21-a.sg

[Dashboard](#) / [My courses](#) / [cs330su21-a.sg](#) / [Week 9 \(5 July - 11 July\)](#) / [Programming Assignment 2 - Divide and Conquer](#)

-  [Description](#)

 [Submission](#)

 [Edit](#)

 Submission view

Grade

Reviewed on Saturday, 17 July 2021, 12:54 AM by Automatic grade
grade: 100.00 / 100.00

Assessment report [-]

Summary of tests

```
+-----+
|  7 tests run/ 7 tests passed  |
+-----+
```

Submitted on Saturday, 17 July 2021, 12:54 AM ([Download](#))
q.cpp

```

1  /*!*****
2  \file    q.cpp
3  \author  Goh Wei Zhe
4  \par    DP email: weizhe.goh@digipen.edu
5  \par    Course: CS330
6  \par    Section: A
7  \par    Programming Assignment #2
8  \date    16-07-2021
9
10 \brief   1. Practise developing Divide and Conquer based algorithm to solve
11           problems.
12
13           2. Practise implementing Partition algorithm and apply it to solve K-th
14           largest element problem, sorting problem, etc.
15 *****/
16
17 // only include the following two head files
18 #include <iostream>
19 #include <vector>
20 // Don't add more
21
22 namespace CS330
23 {
24     namespace divide
25     {
26         // This function is used to print a list
27         void print(std::vector<int> & nums)
28         {
29             for (std::vector<int>::const_iterator i = nums.begin(); i != nums.end();
30                  ++i)
31                 std::cout << *i << ' ';
32             std::cout << std::endl;
33         }
34
35         // This function is used to check your task 3 - rearrange numbers
36         bool check_neg_bef_pos(std::vector<int> & nums)
37         {
38             for (unsigned i=1; i<nums.size(); ++i)
39                 if(nums[i-1]>0 && nums[i]<0) return false;
40             return true;
41         }
42
43         /*!*****
44         \brief
45         Function to rearrange array elements such that every element on the left of
46         the pivot has a lesser or equal value while every element on the right has
47         a larger or equal value than the pivot.
48
49         \param nums
50         A vector array of unsorted integers
51
52         \param begin
53         Starting index in the array
54
55         \param end
56         Last index in the vector array
57
58         \return
59         Returns the index of the pivot's position
60         *****/
61         int partition(std::vector<int>& nums, int begin, int end)
62         {
63             //first element as pivot
64             int pivot = nums[begin];
65             int i = begin, j = end + 1;
66
67             do
68             {
69                 do i++;
70                 while (nums[i] < pivot);
71
72                 do j--;
73                 while (nums[j] > pivot);
74
75                 std::swap(nums[i], nums[j]);
76
77             } while (i < j);
78
79             //undo last swap when i >= j
80             std::swap(nums[i], nums[j]);
81
82             std::swap(nums[begin], nums[j]);
83
84             return j;
85         }
86
87         /*!*****
88         \brief
89         Function to find out the K-th smallest element in a sorted array
90
91         \param nums
92         A vector array of unsorted integers
93
94         \param begin
95         Starting index in the array
96
97         \param end
98         Last index in the vector array
99
100        \param k
101        The K-th smallest element
102
103        \return
104        Returns the value of the K-th smallest element
105        *****/
106        int find_k(std::vector<int>& nums, int begin, int end, int k)
107        {
108            int temp = nums[begin];

```

```

109     int i = begin;
110     int j = end;
111
112     while (i < j)
113     {
114         while ((i < j) && (nums[j] > temp)) j--;
115         nums[i] = nums[j];
116
117         while ((i < j) && (nums[i] <= temp)) i++;
118         nums[j] = nums[i];
119     }
120
121     nums[i] = temp;
122
123     //if equal, i is smallest K-th element, return its value
124     if (i == k - 1)
125         return nums[i];
126     else
127     {
128         // if i < K-1, search for right array
129         if (i < k - 1)
130             return find_k(nums, i + 1, end, k);
131
132         // if i > K-1, search for left array
133         else
134             return find_k(nums, begin, i - 1, k);
135     }
136 }
137
138 /*!*****
139 \brief
140 Function to rearrange elements of a given array of n integer numbers so that
141 all its negative elements precedes all its positive numbers.
142
143 \param nums
144 A vector array of unsorted integers to be sorted
145 *****/
146 void neg_bef_pos(std::vector<int>& nums)
147 {
148     int value = 0;
149     int j = 0;
150
151     for (size_t i = 1; i < nums.size(); ++i)
152     {
153         value = nums[i];
154
155         //if positive, do nothing
156         if (value > 0)
157             continue;
158
159         //if negative, shift elements from 0 - j one index to the right
160         j = i - 1;
161         while (j >= 0 && nums[j] > 0)
162         {
163             nums[j + 1] = nums[j];
164             j--;
165         }
166
167         //place negative value on the right
168         nums[j + 1] = value;
169     }
170 }
171
172 /*!*****
173 \brief
174 Function to partition a vector array using the parameter pivot to rearrange
175 the list.
176
177 \param nums
178 A vector array of unsorted integers
179
180 \param begin
181 Starting index in the array
182
183 \param end
184 Last index in the vector array
185
186 \param pivot
187 Take last element of the bolt as pivot
188
189 \return
190 Returns the partition index of an array based on the pivot element of other
191 array.
192 *****/
193 int partition_pivot(std::vector<int>& nums, int begin, int end, int pivot)
194 {
195     int i = begin;
196
197     for (int j = begin; j < end; j++)
198     {
199         if (nums[j] < pivot)
200         {
201             std::swap(nums[i], nums[j]);
202             i++;
203         }
204         else if (nums[j] == pivot)
205         {
206             std::swap(nums[j], nums[end]);
207             j--;
208         }
209     }
210
211     std::swap(nums[i], nums[end]);
212     return i;
213 }
214
215 /*!*****
216 \brief

```

```
217     Function to match each element of bolt to its nut.
218
219     \param nuts
220     A vector array of unsorted integers
221
222     \param bolts
223     A vector array of unsorted integers
224
225     \param begin
226     Starting index in the array
227
228     \param end
229     Last index in the vector array
230     *****/
231 void nuts_bolts_match(std::vector<int>& nuts, std::vector<int>& bolts,
232 int begin, int end)
233 {
234     if (begin < end)
235     {
236         //choose last character of bolts array for nuts partition.
237         int pivot = partition_pivot(nuts, begin, end, bolts[end]);
238
239         //use partition of nuts to choose for bolt partition.
240         partition_pivot(bolts, begin, end, nuts[pivot]);
241
242         //Recursive function
243         nuts_bolts_match(nuts, bolts, begin, pivot - 1);
244         nuts_bolts_match(nuts, bolts, pivot + 1, end);
245     }
246 }
247 }
248 }
249 }
```

[VPL](#)

◀ Attendance cs330su21-a.sg
Thursday 08/07/2021 11:00am-
12:40pm

Jump to...

⬆

Attendance cs330su21-a.sg Tuesday
13/07/2021 11:00am-12:40pm ▶

You are logged in as [Wei Zhe GOH](#) ([Log out](#))
[cs330su21-a.sg](#)
[Data retention summary](#)
[Get the mobile app](#)