# cs330su21-a.sg

## 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

```cpp
/*!****************************************************************************
\file    q.cpp
\author  Goh Wei Zhe
\par     DP email: weizhe.goh@digipen.edu
\par     Course: CS330
\par     Section: A
\par     Programming Assignment #2
\date    16-07-2021

\brief  1. Practise developing Divide and Conquer based algorithm to solve
           problems.

        2. Practise implementing Partition algorithm and apply it to solve K-th
           largest element problem, sorting problem, etc.
******************************************************************************/

// only include the following two head files
#include <iostream>
#include <vector>
// Don't add more

namespace CS330
{
  namespace divide
  {
    // This function is used to print a list
    void print(std::vector<int> & nums)
    {
      for (std::vector<int>::const_iterator i = nums.begin(); i != nums.end();
          ++i)
        std::cout << *i << ' ';
      std::cout << std::endl;
    }

    // This function is used to check your task 3 - rearrange numbers
    bool check_neg_bef_pos(std::vector<int> & nums)
    {
      for (unsigned i=1; i<nums.size(); ++i)
          if(nums[i-1]>0 && nums[i]<0) return false;
      return true;
    }

    /*!************************************************************************
    \brief
    Function to rearrange array elements such that every element on the left of
    the pivot has a lesser or equal value while every element on the right has
    a larger or equal value than the pivot.

    \param nums
    A vector array of unsorted integers

    \param begin
    Starting index in the array

    \param end
    Last index in the vector array

    \return
    Returns the index of the pivot's position
    **************************************************************************/
    int partition(std::vector<int>& nums, int begin, int end)
    {
        //first element as pivot
        int pivot = nums[begin];
        int i = begin, j = end + 1;

        do
        {
            do  i++;
                while (nums[i] < pivot);

            do  j--;
                while (nums[j] > pivot);

            std::swap(nums[i], nums[j]);

        } while (i < j);

        //undo last swap when  i >= j
        std::swap(nums[i], nums[j]);

        std::swap(nums[begin], nums[j]);

        return j;
    }

    /*!************************************************************************
    \brief
    Function to find out the K-th smallest element in a sorted array

    \param nums
    A vector array of unsorted integers

    \param begin
    Starting index in the array

    \param end
    Last index in the vector array

    \param k
    The K-th smallest element

    \return
    Returns the value of the K-th smallest element
    **************************************************************************/
    int find_k(std::vector<int>& nums, int begin, int end, int k)
    {
        int temp = nums[begin];
        int i = begin;
        int j = end;

        while (i < j)
        {
            while ((i < j) && (nums[j] > temp)) j--;
            nums[i] = nums[j];

            while ((i < j) && (nums[i] <= temp)) i++;
            nums[j] = nums[i];
        }

        nums[i] = temp;

        //if equal, i is smallest K-th element, return its value
        if (i == k - 1)
            return nums[i];
        else
        {
            // if i < K-1, search for right array
            if (i < k - 1)
              return find_k(nums, i + 1, end, k);

            // if i > K-1, search for left array
            else
              return find_k(nums, begin, i - 1, k);
        }
    }

    /*!***************************************************************************
    \brief
    Function to rearrange elements of a given array of n integer numbers so that
    all its negative elements precedes all its positive numbers.

    \param nums
    A vector array of unsorted integers to be sorted
    *****************************************************************************/
    void neg_bef_pos(std::vector<int>& nums)
    {
        int value = 0;
        int j = 0;

        for (size_t i = 1; i < nums.size(); ++i)
        {
            value = nums[i];

            //if positive, do nothing
            if (value > 0)
                continue;

            //if negative, shift elements from 0 - j one index to the right
            j = i - 1;
            while (j >= 0 && nums[j] > 0)
            {
                nums[j + 1] = nums[j];
                j--;
            }

            //place negative value on the right
            nums[j + 1] = value;
        }
    }

    /*!***************************************************************************
    \brief
    Function to partition a vector array using the parameter pivot to rearrange
    the list.

    \param nums
    A vector array of unsorted integers

    \param begin
    Starting index in the array

    \param end
    Last index in the vector array

    \param pivot
    Take last element of the bolt as pivot

    \return
    Returns the partition index of an array based on the pivot element of other
    array.
    *****************************************************************************/
    int partition_pivot(std::vector<int>& nums, int begin, int end, int pivot)
    {
        int i = begin;

        for (int j = begin; j < end; j++)
        {
            if (nums[j] < pivot)
            {
                std::swap(nums[i], nums[j]);
                i++;
            }
            else if (nums[j] == pivot)
            {
                std::swap(nums[j], nums[end]);
                j--;
            }
        }

        std::swap(nums[i], nums[end]);
        return i;
    }

    /*!***************************************************************************
    \brief
```

```
    Function to match each element of bolt to its nut.

    \param nuts
    A vector array of unsorted integers

    \param bolts
    A vector array of unsorted integers

    \param begin
    Starting index in the array

    \param end
    Last index in the vector array
    ***************************************************************************/
    void nuts_bolts_match(std::vector<int>& nuts, std::vector<int>& bolts,
    int begin, int end)
    {
        if (begin < end)
        {
            //choose last character of bolts array for nuts partition.
            int pivot = partition_pivot(nuts, begin, end, bolts[end]);

            //use partition of nuts to choose for bolt partition.
            partition_pivot(bolts, begin, end, nuts[pivot]);

            //Recursive function
            nuts_bolts_match(nuts, bolts, begin, pivot - 1);
            nuts_bolts_match(nuts, bolts, pivot + 1, end);
        }
    }
  }
}
```

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 ►