Lab Assignment #10

Purpose: Recursive Function
Collaboration: You have to individually.

Due is on Sunday 11:59 PM, November 29, 2015

Quick Sort

Implement the “quick sort” function.

Sorting a list of data is a very common task we use computer to do. For example, I was recently looking for apartments near the MSU campus on the website rent.com. When I search “apartments near MSU” on the website, it will return me a list of apartments and allow me to sort these apartments by price, distance, or rating, so that I can easily pick from them. Writing an efficient sorting program (for example the “quick sort” you will be working on in this assignment) is also a common job interview problem for programmers.

You need to write a C function to sort a given one-dimensional array of integers, using the “quick sort” algorithm. Quick sort is a divide and conquer algorithm. It first divides a large array into two smaller sub-arrays and then recursively sort the sub-arrays.

When you define a quick sort function, it should take three inputs (parameters), e.g.,
void quickSort(int A[], int low, int high)

A is the array to be sorted, low and high are two indices that specify a part (sub-array) of the array. For example, if low is 2 and high is 4, it specifies the part of the array from A[2] to A[4]. Each call of the quickSort function actually sorts the sub-array in A specified by low and high. To sort the entire array, we just call the quickSort function like this (suppose the size of the array A is 7):

quickSort(A, 0, 6);

Quick sort works in this way:
Step 1: Pick an element, called a pivot, from the (sub)array. In our case, we will always pick the element whose index is

low + (high - low) / 2

as the pivot.
Step 2: Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it. After this step, the pivot is already in its final position (of the finally sorted array). This step is called the partition step, because it partitions the array into two sub-arrays.
Step 3: Recursively apply the above steps to the sub-array of elements with the smaller values and (separately) to the sub-array of elements with the greater values.
The second step (partition) is the most tricky step. It can be done in different ways. For this homework, you need to implement this step as described by the following pseudo-code. 

**Pseudo-code for the partition step of quick sort:**

What are given: the array A, the low and high indices, and the pivot index

**Procedure:**

```
pivotValue = A[pivotIndex]
  // put the chosen pivot at A[high]
  swap A[pivotIndex] and A[high]
  // Initially, the final position (after sorting) of the pivot is set to the low index
  pivotFinalPos = low
  // Compare remaining array elements against pivotValue (= A[high])
  for i from low to (high - 1):
    if A[i] < pivotValue
      swap A[i] and A[pivotFinalPos]
      pivotFinalPos = pivotFinalPos + 1
    end if
  end for
  // Move pivot to its final place
  swap A[pivotFinalPos] and A[high]
```

2.1 (1 point) As we can see, quick sort uses recursion. Explain what is the recursive pattern, i.e., what is the smaller problem(s) contained in a big problem?

2.2 (1 points) What should be the termination condition for quick sort, i.e., when should quick sort stop calling itself?

2.3 (2 points) Write a pseudo-code for the quick sort function at the high level (You do not need to write the pseudo-code for the “partition” sub-routine since it is already given). You can write the pseudo-code in your own way and do not need to follow a convention, as long as the logic is correct and understandable to us.

2.4 (4 points) Implement the quick sort function and test it with a given example array. You only need to implement the quicksort() function as dfa bellow belows :

Your program should look like this:

```c
#include <stdio.h>
void quickSort(int a[], int low, int high) {
  /* Your code goes here */
}
```
int main() {
    int a[7] = {6, 12, 5, 97, 1, 8, 56};

    quickSort(a, 0, 6);

    printf("The sorted array is: ");
    int i;
    for(i = 0; i < 7; i++)
        printf("%d ", a[i]);
    printf("\n");
}

Lab Submission
For the questions that you need to write a code, copy and paste your whole code as the answer. The deliverable for this assignment is the following file:

    Lab10.c

Be sure to use the specified file name ("lab10.c") and format. To submit your homework, you should upload your file to handin system using the following link:

    http://secure.cse.msu.edu/handin