Assignment 5

CS270 Spring 2017

Due April 17, 2017 at 11:59pm

1 Recursive Sort

Write a MIPS program that uses an implementation of quicksort to sort an array of numbers. The values of the array will be given to you via standard input and separated by newlines. The first number will be the number of elements in the array and followed by the elements of the array in order. You can assume that there will be at least one, and no more than 20 numbers in the array. Each number will fit in a signed 4 byte integer. After sorting the numbers, you should write them to standard out in increasing order. Your code should be able to handle an array with numbers that have the same value. For full credit, your code must be a valid translation of the quicksort implementation on the next page of this document. Your code must adhere to the MIPS style guidelines discussed in class. Your code must include the appropriate methods to perform the sorting. The array should live in the static data portion of memory.

Sample Input

```
7
2
5
1
-7
2
4
16
```

Sample Output

```
The elements sorted in ascending order are: -7, 1, 2, 2, 4, 5, 16
```

2 Submission Details

You will submit a **single .asm file** of your source code to Autolab. You may submit your code to Autolab multiple times. It will run it against some test cases I have provided. When I grade your work, I will grade it based on the performance on these public tests, as well as my own additional tests.

This is an **individual** assignment. Your code should adhere to the MIPS style guidelines from the notes. Your code should assemble and run without crashing. You will receive a 0 if the code does not assemble or crashes.

3 Grading Guidelines

You will be assessed on the correctness of your code, the quality of your solution, and adherence to the style guidelines. This assignment is worth 100 points.

Quicksort Implementation - C

```
int partition(int arr[], int left, int right) {
      int i = left, j = right;
      int tmp;
      int pivot = arr[(left + right) / 2];

      while (i <= j) {
            while (arr[i] < pivot)
                  i++;

            while (arr[j] > pivot)
                  j--;

            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;
                  j--;
            }
      }

      return i;
}

void quickSort(int arr[], int left, int right) {
      int index = partition(arr, left, right);

      if (left < index - 1)
            quickSort(arr, left, index - 1);

      if (index < right)
            quickSort(arr, index, right);
}
```