CS101 – Computer Programming
Quiz for Tuesday Batch – 30 September 2014

Q1. Consider the following variation of function '**mergeSort**' to sort the array in descending order. In the version of 'mergeSort' discussed in class, the array is divided into two sub arrays of (almost) equal size. In the following, the array is divided into **three** sub arrays of almost equal size

The function '**mergeSortedThreeSubArrays(A, start, mid1, mid2, end)**' takes in input array A, where A[start] to A[mid1-1] **and** A[mid1] to A[mid2-1] **and** A[mid2] to A[end-1] are each sorted in descending order and when returning merges the elements from A[start], ..., A[end-1] in descending order.

The function '**selectionSort(A, start, end)**' sorts the array 'A' from 'start' to 'end-1' in descending order.

```
void mergeSort(int A[], int start, int end) {
      if(end == start)
            return;
      if(end – start < 3) {
            selectionSort(A, start, end);
            return;
      }
      int mid1 = start + (end – start)/3;
      int mid2 = start + 2*(end - start)/3;
      mergeSort(A, start, mid1);
      mergeSort(A, mid1, mid2);
      mergeSort(A, mid2, end);

      mergeSortedThreeSubArrays(A, start, mid1, mid2, end);
      return;
}

int main() {
      int A[20];
      //Code to take input for 20 elements
      mergeSort(A, 0, 20);
      return 0;
}
```

The total number of calls to the function 'mergeSort'(including the one from the 'main' program) when the above program executes is:
(Select one of the choices)

A. 17
B. 16
C. 15
D. 18
E. None of the above

Refer Q2 on the next page

Q2. Consider the following recursive implementation of '**binarySearch**' algorithm:

```
int binarySearch(int A[], int start, int end, int ele) {
      int mid = (start + end)/2;
      if(start == end-1) {
            if(!compareElements(ele, A[mid])) {
                  return mid;
            } else {
                  return -1;
            }
      }
      if(!compareElements(ele, A[mid])) {
            return mid;
      } else if (compareElements(ele, A[mid]) < 0) {
            return binarySearch(A, start, mid, ele);
      } else {
            return binarySearch(A, mid, end, ele);
      }
}

int main() {
 int A[20];
 int ele;
 //Code to take input for 20 elements and the element to be searched in “ele”
 int index = binarySearch(A, 0, 20, ele);
 return 0;
}
```

Note: The search is said to be **successful** if it returns the index of the searched element (if it exists in the array), and returns -1 if it doesn't exist in the array. Now, consider the following statements about the given program and choose the correct option(s).

A. If the array A is sorted in ascending order then using the following definition of function '`compareElements`, the search will be successful.

```
int compareElements(int a, int b) {
      if( a > b ) {
            return -1;
      } else {
            return 1;
      }
}
```

B. If the array 'A' is sorted in descending order then using the following definition of function '`compareElements'`, the search will be successful

```
int compareElements(int a, int b) {
      if( a == b ) {
            return 0;
      } else if ( a > b ) {
            return 1;
      } else {
            return -1;
      }
}
```

C. If the array A is sorted in ascending order then using the following definition of function '`compareElements`', the search will be successful

```
int compareElements(int a, int b) {
      if( a == b ) {
            return 0;
      } else if ( a > b ) {
            return 1;
      } else {
            return -1;
      }
}
```

D. If the array A is sorted in descending order then using the following definition of function '`compareElements`', the search will be successful

```
int compareElements(int a, int b) {
      return (b - a);
}
```

E. None of the above