Lab 9: Sorting

In this lab, you are going to implement three different sorting algorithms to sort an array of 50,000 random integers. You should measure the elapsed time for each method. Repeat the experiment on the sorted array as well and compare the elapsed time. You may include the library and use the following sample code to measure the running time of your function.

```
const int CLOCKS_PER_MS = CLOCKS_PER_SEC/1000; //clock per mi
lliseconds
...
clock_t Start = clock();
//call sort function here
clock_t End = clock();
int elapsedTime = (End - Start)/CLOCKS_PER_MS; // converts el
apsed time from microseconds to milliseconds.
```

Implement the following function to sort an array of integers:

- `void Quicksort_midpoint(int numbers[], int i, int k):` this function sorts the given array in the range from i to k using quicksort method. In this function, pivot is the midpoint element (numbers[(i+k)/2]). (implementation of this function is given in section 21.5)
- `void Quicksort_medianOfThree(int numbers[], int i, int k):` this function utilizes different pivot selection technique in quicksort algorithm. The pivot is the median of the following three values: leftmost (numbers[i]), midpoint (numbers[(i+k)/2]) and rightmost (numbers[k]). You should modify the partition function given in section 21.5 to select the pivot using median-of-three technique.
- `void InsertionSort(int numbers[], int numbersSize):` this function sorts the given array using InsertionSort method. (implementation of this method is provided in section 21.3).

You can use the following function to generate three similar array of random integers.

```
const int NUMBERS_SIZE = 50000;
...
int genRandInt(int low, int high) {
   return low + rand() % (high - low + 1);
}
void fillArrays(int arr1[], int arr2[],int arr3[]){
    for(int i = 0; i < NUMBERS_SIZE; ++i){
        arr1[i] = genRandInt(0,NUMBERS_SIZE);
        arr2[i] = arr1[i];
        arr3[i] = arr1[i];
    }
}
```