TUTORIAL 11

ECE 532

1. List down analysis of performance comparison between searching method using linear and Binary search tree.

2. Mark the following statements as true or false.

a. A sequential search of a list assumes that the list is in ascending order.

b. A binary search of a list assumes that the list is sorted.

c. A binary search is faster on ordered lists and slower on unordered lists.

d. A binary search is faster on large lists, but a sequential search is faster on small lists.

3 Sort the sequence 3, 1, 4, 1, 5, 9, 2, 6, 5 using insertion sort.

4 Sort 3, 1, 4, 1, 5, 9, 2, 6 using mergesort.

5. Sort the following list using selection sort as discussed in this chapter. Show the list after each iteration of the outer for loop.

26, 45, 17, 65, 33, 55, 12, 18

6. Assume the following list of keys: 5, 18, 21, 10, 55, 20 The first three keys are in order. To move 10 to its proper position using insertion sort as described in this chapter, exactly how many key comparisons are executed?

7. Assume the following list of keys: 28, 18, 21, 10, 25, 30, 12, 71, 32, 58, 15 This list is to be sorted using insertion sort as described in this chapter for array-based lists. Show the resulting list after six passes of the sorting phase—that is, after six iterations of the for loop.

8. Both mergesort and quicksort sort a list by partitioning the list. Explain how mergesort differs from quicksort in partitioning the list

9. Quicksort algorithm is applied on a set of integers {35,50,40,20,55,45} given in **Figure 1**. Trace the movement of the data at the end of each partition operation in the table format below. Assume the pivot value is the first element in the array.

STEP	DATA					
0	**35**	**50**	**40**	**20**	**55**	**45**
1						
2						
3						

Figure 1: Array of Integers

10. **Figure 2** shows a bubble sort program. Analyse and write the movement of the elements of the array for every step of the sorting operations in the table format as shown in **Table 1.**

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int nums[5] = {78,90,68,10,37};
    int Swap;
    cout <<"Data items in original order\n";

    for(int ctr=0; ctr<5; ctr++)
      {
        cout<< setw(4) <<nums[ctr];
      }
    for(int i=0; i<5; i++)
        for(int ii=0; ii<5; ii++)
            if (nums[ii] > nums[ii + 1])
                {
                    Swap = nums[ii];
                    nums[ii] = nums[ii + 1];
                    nums[ii + 1] = Swap;
                }
    cout<<"Data items in ascending order\n";
    for (int iii=0; iii<5; iii++)
        cout<< setw(4) << nums[iii];
    cout<< endl <<endl;

}
```

Figure 2

Table 1

Original sequence	Data sequence when i =0	Data sequence when i =1	…	Data sequence when i = 4
78				
90				
68				
10				
37				