Usman Institute of Technology

Department of Computer Science – Fall 2018

CS-211 Data Structures and Algorithms
Lab Manual

OBJECTIVE:

1. *Understand and implement Recursion and Quick Sort.*

Name : _____

Roll No. : _____

Semester : _____ Section: _____

Date : _____

Remarks : _____

Signature : _____

Lab 06: Understand and implement Recursion and Quick Sort.

1. **Recursive Problems**
   a. Create a class RecursiveOP in order to implement Recursive functions.

   b. Create a recursive function GetFactorail which takes an integer number as input and returns the Factorial of the given number

   int GetFactorail (int num)

   c. Create a recursive function CountNOD which takes an integer number as input and returns the count of the number for the given number

   int CountNOD (int num)

   d. Create a recursive function BaseConverter which takes an integer number as input and returns the equivalent of that number in Base2.

   int BaseConverter (int num)

   e. Create a recursive function GetGCD which takes two integer numbers as input and returns their greatest common divisor (GCD).

   int GetGCD (int num1, int num2)

2. **Quick Sort**
   a. Create a function GenerateRandomNum which takes the size of the array as input, generate 'n' random numbers in an integer array and returns that array.

   int[] GenerateRandomNum(int n)

   b. Create a recursive function to sort the array generated in the above task using Quicksort and returns the sorted array.

   int[] Quicksort (int arr[])

Algorithm: QuickSort (Data, Start, End):

Step 1 If (Start < End)
Step 2:Set PINDEX = Call Partition (Data, Start, End)
Step 3 Call QuickSort (Data, Start, PINDEX-1)
Step 4 Call QuickSort (Data, PINDEX + 1, End)

Algorithm: Partition(Data, Start, End) :

Step 1 Set PIVOT:= Data[End]
Step 2 Set PINDEX = Start -1
Step 3 Repeat For (Start to End -1)
Step 4 If (Data[i] <= PIVOT
Step 5 Swap (Data[i], Data[PINDEX])
Step 6 PINDEX: = PINDEX +1
Step 7 End if
Step 8 End for
Step 9 Swap (Data[PINDEX], Data[End])
Step 10 Return PINDEX

c. Repeat task 2 (b) and sort the array in descending order.

int[] QuicksortDesc (int arr[])

d. Create a function which compares the execution time of the functions created in Task 2(b) and 2(c) for n number of elements.

Void Timer(int n)

The output for n = 100 should be look like:

Time comparison for 100 numbers:
Quick Sort in Ascending Order: x seconds
Quick Sort in Descending Order: y seconds