Name: ____________________ Reg #: ____________________ Section: ______________

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Spring 2018**
Duration:	**60 Minutes**	**Total Marks:**	**40**
Paper Date:	**27-Feb-18**	**Weight**	**15%**
Section:	**ALL**	**Page(s):**	**6**
Exam:	**Midterm 1 Solution**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1-5	6	7	8	Total
Marks	/ 14	/ 10	/ 6	/10	/ 40

40

Q1) True / False. Justify your answer [4 Marks]

1. $3^{4+n} = \Theta(3^n)$ --------True-----

   $3^{4+n} = 3^4 \cdot 3^n = O(n)$ Omega (n)

   $3^4 \cdot 3^n = \Omega(n)$

2. $n \lg n = \Theta(3n\log_8 n)$ ----False--------

$3n\log_8 n = 3n\lg n / \lg 8 = 3n \lg n /3 = n \lg n$
$n^3 \lg n = \Omega(n \lg n)$
$n^3 \lg n \neq O(n \lg n)$

Q2) Which of the following sort algorithms are stable? [2 Marks]

a) Quick Sort
b) Merge Sort
c) Insertion Sort
d) Count Sort

Answer: b, c, and d

Q3) Which of the following sort algorithms are guaranteed to be O(n log n) even in the worst case? [2 Mark]

e) Quick Sort
f) Merge Sort
g) Insertion Sort

Answer: Merge Sort

Q4) Given that Merge sort's worst case time is better than Quick sort's, why bother ever to use Quick sort at all? (i.e., why is Quick sort so commonly used in practice?) [4 Marks]

Probability of Quick sort running in n^2 time is very low, it runs in nlg n time with high probability.
Quick sort is in place whereas Merge sort is not in place.

Q5) Suppose we are sorting an array of eight integers using Quick sort, and we have just finished the first partitioning with the array looking like this:
2 5 1 7 9 12 11 10
Which statement is correct? [2 Mark]

a) The pivot could be either the 7 or the 9.
b) The pivot could be the 7, but it is not the 9.
c) The pivot is not the 7, but it could be the 9.
d) Neither the 7 nor the 9 is the pivot.

Answer: a

Q6) A d-***ary heap*** is like a binary heap, but (with one possible exception) non-leaf nodes have d children instead of 2 children.
a) What is the height of a d-ary heap of n elements in terms of n and d? [2 Mark]

Answer: $\log_d n$

b) Give an efficient implementation of EXTRACT-MIN in a d-ary min-heap. Write pseudo code. You do not need to define functions for getChildren(node i) and getParent(node i). You can assume they are given to you. [4 Marks]

Answer:

```
ExtractMin(int [] Heap, int size )
{
    int min = heap[1]
    swap (heap [1], heap[size])
    size = size -1
    curr = 1
    while ( curr < size )
     {

            childArray = GetChildren(curr)
            minChild = getMinimum(childArray)
            If(heap[curr] > heap[minChild])
           {
                Swap (heap[curr], heap[minChild])
                Curr = minChild;
            }
           Else
                Break;

     }
     Return min
 }
```

c) Analyze its running time of EXTRACT-MIN defined in above question in terms of d and n. [4 Marks]

Answer: $d \log_d n$

Q7) Use a recursion tree to determine a good asymptotic upper bound on the recurrence [6 Marks]
T (n) = 4 T (n/2) + n.

Solution

Height of tree = lg (n)

Running time = n + 2n + 4n + ...

$$= \sum_{i=0}^{lg\,n} 2^i n$$

$$= n \sum_{i=0}^{lg\,n} 2^i$$

$$= n (1 - 2^{lgn})$$

$$= n (1 - n)$$

$$= n - n^2$$

$$= O (n^2)$$

Q8) Given two sorted arrays X[ ] and Y[ ] of sizes M and N where M ≥ N, devise an algorithm to merge them into a new sorted array C[ ] using O(N lg M) comparison operations. Suppose arrays M and N are indexed from 1 to M and from 1 to N respectively. [10 Marks]

Hint: use binary search.

Solution

```
Merge (X, Y, M, N )
{
     Int j=0;
     int ind = 0;
     for (int i=0; i< N,  i++)
    {
          Int temp = binarySearch(X, Y[i])
          // temp contains index of last element less than Y[i] in X
         While (j < temp)
                C[ind++] = X [j++]
         C[ind++] = Y[i]
     }
}
```