Maharishi International University (MIU)

MIDTERM

Course Title and Code: *CS 435 - Design and Analysis of Algorithms*
Instructor: *Dr. Emdad Khan*
Date: *Friday 07/30/2021*
Duration: *9am – 11:30 pm*

Student Name:

Student ID:

Total Mark
_____
46

1. This is a closed book exam. Do not use any notes or books!
2. Show your work. Partial credit will be given. Grading will be based on correctness, clarity and neatness.
3. We suggest you to read the whole exam before beginning to work on any problem.
4. There are 4 questions worth a total of 46 points, on 9 pages (including this one)
5. You can use all back pages for scratch paper (may use for answers if you have used up the designated space for answer).
6. You can use a basic calculator (No smart phone unless network is disabled).

Question 1: 10 points (3 + 4 + 3)

a. Show the running time of the following code using big O notation (*show your work*):

```
for (int i = 0; i < n + 100; ++i)
{
      for (int j = 0; j < i * n ; ++j)
      {
            sum = sum + j;
      }

      for (int k = 0; k < n + n + n; ++k)
      {
            c[k] = c[k] + sum;
      }
}
```

Ans.

b. Determine whether f is O(g), Theta (g) or Omega (g). Mention all that applies. Show your work.

i) $f = 2^{2n}$, $g = 2^n$

ii) $f = n^{\lg m}$ and $g = m^{\lg n}$; **Show your reasoning / work.**

c. Give a Big O estimate for the following function

$$(2^n + n^2)(n^3 + 3^n).$$

Question 2: 12 points (4 + 4 + 4)

For each of the following recurrences, derive an expression for the running time using iterative, substitution or Master Theorem.

a. Consider the following recurrence algorithm **[Use Master Theorem – See LAST Page]**

```
long power(long x, long n)
  if (n==0) return 1;
  if (n==1) return x;
  if ((n % 2) == 0)
    return power(x*x, n/2);
  else
    return power(x*x, n/2) * x;
```

Assume n is power of 2.

i. (2 points) Write a recurrence equation for T(n)

ii. (2 points) Solve recurrence equation using Master's method i.e. give an expression for the runtime T(n).

b. Use **Iterative method**

$T(n) = T\left(\frac{n}{2}\right) + T(n-2) + c \quad for\ n > 0$ [Use Iterative method]

T(n) = 1 for n = 0.

Question 2: (continued)

c. Use Induction to show that

$$\sum_{r=1}^{n} r(r+1) = \frac{1}{3}n(n+1)(n+2)$$

Question 3: 11 points (4 + 4 + 3)

a. (a) Assume you are creating an array data structure that has a fixed size of n. You want to backup this array after every so many insertion operations. Unfortunately, the backup operation is quite expensive, it takes n time to do the backup. Insertions without a backup just take 1 time unit.

(i) How frequently can you do a backup and still guarantee that the amortized cost of insertion is O(1)?

(ii) Prove that you can do backups in O(1) amortized time. Use the accounting method for your proof.

(b) Use the QuickSelect algorithm to manually compute the 5th smallest element of the array [1, 5, 23, 0, 8, 4, 33]. Assume that the rightmost element is used as the pivot in each case. Show what happens in each self-call, indicating the new input array and the current value of *k*.

Question 3: (continued)

c. Use RadixSort, with two bucket arrays and radix = 12, to sort the following array: [63, 1, 48, 53, 24, 10, 12, 30, 100, 141, 17]. Show all steps of the sorting procedure. Then explain why the running time is O(n). What would be the running time if you used ONE bucket?
What would be the running time if you used ONE bucket?

Question 4: 13 points (4 + 5 + 4)

a. What is the worst case running time of Quicksort? Can you improve the worst case running time of quicksort? If so, describe to what value and how.

b. (i) 2 points - Explain with an example what is meant by "Mathematics is not sound".

(ii) **3 points** - Show how Quicksort is not stable by using in-place random partitioning algorithm and the following 4 numbers {4a, 4b, 4c, 4d} (show all steps).

Question 4: (continued)

c. You would like to determine which of your Facebook friends are early adopters. So, you have decided to sort them using Facebook account ids which are 64-bit numbers. Which sorting algorithm will be most appropriate – **Insertion sort, Merge sort, Quicksort, Counting sort or Radix sort? Explain why.**

Master Formulae

For recurrences that arise from Divide-And-Conquer algorithms (like Binary Search), there is a general formula that can be used.

Theorem. Suppose $T(n)$ satisfies

$$T(n) = \begin{cases} d & \text{if } n = 1 \\ aT(\lceil \frac{n}{b} \rceil) + cn^k & \text{otherwise} \end{cases}$$

where k is a nonnegative integer and a, b, c, d are constants with $a > 0, b > 1, c > 0, d \geq 0$. Then

$$T(n) = \begin{cases} \Theta(n^k) & \text{if } a < b^k \\ \Theta(n^k \log n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$