A

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam — Semester: Spring 2025
Duration: 1 Hour 30 Minutes — Full Marks: 30

CSE 221: Algorithms

Answer the following questions.
Figures in the right margin indicate marks.

Name:	ID:	Section:

1 **a.** **CO2** Analyze the following code snippet and **express** the time complexity using the Big O-notation. You must mention the time complexity for each loop. **04**

```
int b = 0;
for (int i = n; i > 0; i -= 2) {
    for (int j = n; j > 0; j--) {
        b += i * j;
    }
}

for (int k = 1; k < n; k *= 4) {
    for (int m = 0; m < n; m += 3) {
        if (m == 10) {
            break;
        }
        b += k + m;
    }
 }
 return b;
```

b. **CO2** You are given the following recurrence relation: **02**

$$T(n) = 3T\left(\frac{n}{4}\right) + \Theta(3n^2)$$

Solve the recurrence relation using an appropriate method

c. **CO2** **Prove** whether $5n^2 + 4n = \Theta(T(n))$, where T(n) is the answer you found in part Q1(b). **03**

d. **CO2** Why do we ignore constants and lower-order terms when expressing the time complexity of an algorithm? **Briefly** explain in your own words. **01**

2 Consider a farm with **n** cows of two colors, say, white cows and black cows. The cows are arranged in their barn in such a way that all white cows are kept first, followed by all black cows. Each of the cows is assigned a serial number. **You are to find the serial number of the first black cow.**

a. CO1 **Describe** how to convert the given information into an array of integers. **01**

b. CO1 Given the array of integers from Q(2a), **present** a method with a time complexity **less than O(n)** to solve the mentioned problem in the paragraph. Use pseudocode/ executable code/ step-by-step logical instructions to show your method. **01**

c. CO1 Suppose we implement Quicksort using a strategy where we choose the **pivot as the median of the first, last, and middle elements of a subarray**. To be specific, the pivot is selected as follows: **03**

$$\text{pivot} = \text{median}(A[\text{st}], A[\text{end}], A[\text{mid}]) \quad \text{where} \quad \text{mid} = \left\lfloor \frac{\text{st} + \text{end}}{2} \right\rfloor$$

For example, consider the array A = [6, 2, 9, 1, 7, 3, 8].
In the initial Quicksort call, start = 0 gives A[0] = 6, end = 6 gives A[6] = 8, and mid index = $\frac{(0+6)}{2}$ = 3 gives A[3] = 1. Among these three values—1, 6, and 8—the sorted order is (1, 6, 8), so the median, and thus the chosen pivot, is 6.

What is the worst-case input for this pivot selection strategy in Quicksort? **Generate** such a worst-case scenario with a 7-element array containing the integers 1 through 7, each used exactly once. Briefly explain why it leads to worst-case behavior.

Now consider a factory where **n** humanoid robots work. You are to sort the robots based on some criteria. Answer Q2(d) – (e) based on this.

d. CO1 Each robot can form connections with some of the other robots. That means each robot can form at most (n-1) connections. Which algorithm will be the fastest to sort the robots based on the number of connections they formed? **Write** the name and time complexity of the algorithm. **02**

e. CO1 Suppose, in the factory, there exists a system that can find the tallest robot in O(1) time. That means you are given a function, $find\_tallest(A)$, which returns the tallest robot from a list called **A**, in constant time. **03**

Considering this information, design an algorithm to sort the robots in descending order of their heights. **The time complexity of your algorithm must not exceed O(n).**

i. **Present** your algorithm using pseudocode/ executable code/ step-by-step logical instructions.
ii. **Write** the time complexity of your algorithm.

3 In Rivendell, **n** voters are casting their votes for various political parties. The votes are recorded in an array, where each number represents a party's ID.

However, Rivendell follows a special rule: a party can only form the government if it receives more than $\lfloor \frac{n}{2} \rfloor$, that is, a strict majority. Otherwise, the election results in a deadlock, and a re-election must be held. For example, in the array [1,2,2,2,3,2,1], party 2 wins the majority since it got 4 votes (more than $\lfloor \frac{7}{2} \rfloor = 3$).

a. **CO1** Can two different parties each secure a majority in any given vote array? **Explain** why or why not. **02**

b. **CO1** Given a vote array and a party ID, we need to determine whether the party has won the majority in the array. **Propose** an O(N) algorithm to verify this, where N is the number of voters. **02**

c. **CO1** Now, we want to apply a divide and conquer approach to find the majority from a given vote array. Like merge sort, if we divide the vote array into two halves, can the majority element in each half be different from the majority of the entire array? **Justify** your answer with an example. **02**

d. **CO1** Reuse your algorithm of Q3(b) to complete the following algorithm to find the majority party from the vote array. The algorithm should return -1 if there's no majority party. You can use either pseudo-code or step-by-step instructions. **04**

```
function majority(arr):
     if length(arr) == 1:
          return arr[0]

     mid = length(arr) // 2
     left_majority = majority(arr[0 ... mid])
     right_majority = majority(arr[mid+1 ... length(arr)-1])

     # Merge Step: Reuse your algorithm of Q3(b) to complete
```

B

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam
Duration: 1 Hour 30 Minutes

Semester: Spring 2025
Full Marks: 30

CSE 221: Algorithms

Answer the following questions.
Figures in the right margin indicate marks.

Name:	ID:	Section:

1 **a.** **CO2** Analyze the following code snippet and **express** the time complexity using the Big O-notation. You must mention the time complexity for each loop. **04**

```
int b = 0;
for (int i = n; i > 0; i -= 3) {
    for (int j = 0; j < n; j += 3) {
        if (j == 50) {
            break;
        }
        b += i + j;
    }
}

for (int k = 1; k < n; k *= 3) {
    for (int m = n; m > n; m /= 5) {
        b += k + m;
    }
}
return b;
```

b. **CO2** You are given the following recurrence relation: **02**

$$T(n) = 5T\left(\frac{n}{6}\right) + \Theta(3n)$$

Solve the recurrence relation using an appropriate method

c. **CO2** **Prove** whether $5n + 4 = \Theta(T(n))$, where T(n) is the answer you found in part Q1(b). **03**

d. **CO2** Why do we ignore constants and lower-order terms when expressing the time complexity of an algorithm? **Briefly** explain in your own words. **01**

2 Consider a farm with **n** sheep of two colors, say, white sheep and black sheep. The sheep are arranged in their barn in such a way that all white sheep are kept first, followed by all black sheep. Each of the sheep is assigned a serial number. **You are to find the serial number of the first black sheep.**

a. CO1 **Describe** how to convert the given information into an array of integers. **01**

b. CO1 Given the array of integers from Q(2a), **present** a method with a time complexity **less than O(n)** to solve the mentioned problem in the paragraph. Use pseudocode/ executable code/ step-by-step logical instructions to show your method. **01**

c. CO1 Suppose we implement Quicksort using a strategy where we choose the **pivot as the median of the first, last, and middle elements of a subarray**. To be specific, the pivot is selected as follows: **03**

$$\text{pivot} = \text{median}(A[\text{st}], A[\text{end}], A[\text{mid}]) \quad \text{where} \quad \text{mid} = \left\lfloor \frac{\text{st} + \text{end}}{2} \right\rfloor$$

For example, consider the array A = [6, 2, 9, 1, 7, 3, 8].
In the initial Quicksort call, start = 0 gives A[0] = 6, end = 6 gives A[6] = 8, and mid index = $\frac{(0+6)}{2}$ = 3 gives A[3] = 1. Among these three values—1, 6, and 8—the sorted order is (1, 6, 8), so the median, and thus the chosen pivot, is 6.

What is the worst-case input for this pivot selection strategy in Quicksort? **Generate** such a worst-case scenario with a 7-element array containing the integers 1 through 7, each used exactly once. Briefly explain why it leads to worst-case behavior.

Now consider a factory where **n** humanoid robots work. You are to sort the robots based on some criteria. Answer Q2(d) – (e) based on this.

d. CO1 Each robot can form connections with some of the other robots. That means each robot can form at most (n-1) connections. Which algorithm will be the fastest to sort the robots based on the number of connections they formed? **Write** the name and time complexity of the algorithm. **02**

e. CO1 Suppose, in the factory, there exists a system that can find the tallest robot in O(1) time. That means you are given a function, $find\_tallest(A)$, which returns the tallest robot from a list called **A**, in constant time. **03**

Considering this information, design an algorithm to sort the robots in descending order of their heights. **The time complexity of your algorithm must not exceed O(n).**

i. **Present** your algorithm using pseudocode/ executable code/ step-by-step logical instructions.
ii. **Write** the time complexity of your algorithm.

3 In Transylvania, **n** audiences are casting their votes for various movies. The votes are recorded in an array, where each number represents a movie's ID.

The theater has a rule: for a movie to be selected for screening, it must receive more than $\lfloor \frac{n}{2} \rfloor$, that is, a strict majority. Otherwise, the theatre rejects this voting, and a re-election must be held. For example, in the array [1,2,2,2,3,2,1], movie 2 wins the majority since it got 4 votes (more than $\lfloor \frac{7}{2} \rfloor = 3$).

a. CO1 Can two different movies each secure a majority in any given vote array? **Explain** why or why not. **02**

b. CO1 Given a vote array and a movie ID, we need to determine whether the movie has won the majority in the array. **Propose** an O(N) algorithm to verify this, where N is the number of audiences. **02**

c. CO1 Now, we want to apply a divide and conquer approach to find the majority from a given vote array. Like merge sort, if we divide the vote array into two halves, can the majority element in each half be different from the majority of the entire array? **Justify** your answer with an example. **02**

d. CO1 Reuse your algorithm of Q3(b) to complete the following algorithm to find the majority movie from the vote array. The algorithm should return -1 if there's no majority movie. You can use either pseudo-code or step-by-step instructions. **04**

```
function majority(arr):
     if length(arr) == 1:
          return arr[0]

     mid = length(arr) // 2
     left_majority = majority(arr[0 ... mid])
     right_majority = majority(arr[mid+1 ... length(arr)-1])

     # Merge Step: Reuse your algorithm of Q3(b) to complete
```