A

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam | Semester: Fall 2024
Duration: 1 Hour 15 Minutes | Full Marks: 30

CSE 221: Algorithms

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

Name:	ID:	Section:

1 a. **CO2** Consider the following code snippet to answer questions (i) and (ii).

```
1.  Function F(n):
2.      If (n == 1):
3.          Return True;
4.      Else:
5.          X = F(n/2);
6.          Y = F(n/2);
7.          Z = F(n/2);
8.          U = A(n);
9.
10. Function A(n):
11.     For i := n to 1; i--:
12.         Return True;
```

i. **Write** the time complexity of the function "F" using a recurrence relation. **02**
ii. **Express** the recurrence relation from question (i) with asymptotic upper bound (Big-O) using any convenient method. Show your work. **02**

b. Given an array of **N** elements, you are required to find the **peak element** in it. An element is considered to be a peak if its value is greater than the value(s) of its adjacent element(s) i.e. immediate left and immediate right. If there is only one adjacent element, we take only that in consideration. And the array will contain **exactly one** peak element.

Now answer the following questions.

CO1 i. **Determine** whether each of the following arrays matches the above stem-mentioned conditions: **02**
1. [1, 5, 4, 3, 2]
2. [1, 3, 2, 5, 4]
3. [5, 2, 1, 3, 4]
4. [1, 2, 3, 4, 5]
[N.B.: For each array, it is enough to write only True/False in the answer script.]

CO3 ii. **Present** an algorithm with a time complexity lower than **O(N)** to solve the stem-mentioned problem using pseudocode/ programmable code/ step-by-step logical instructions. **03**

CO2 iii. If the array contains **more than one** peak element, will the presented algorithm in question (ii) still be able to find any peak element? **Explain** briefly. **01**

2 **a.** **CO1** You are designing a computerized logistics system for a warehouse that handles thousands of packages daily. Each package is assigned a unique package ID and a delivery deadline (in terms of 'remaining hours' from the current time) and the packages are initially sorted according to their package_id. The warehouse management system needs to sort the packages by their delivery deadlines in ascending order to prioritize the earliest deliveries. **05**

But there are some conditions to develop this system. The system does not have any issues related to space/memory but it needs the list to be sorted with time complexity $O(nlogn)$ where n is the number of packages to process. Moreover, if any two packages have the same delivery deadline, the package with smaller package_id comes before i.e the algorithm must be stable.

Write the name of the appropriate algorithm for this case and show the solution with pseudocode/programmable code/step-by-step logical instructions.

b. [Continues on the previous question…] After some days, the warehouse noticed a massive increase in daily orders and the system was facing memory issues to hold these massive orders. So they consulted you again. You suggested to them an algorithm which does not require any additional memory. But you also warned them that it can degrade to the time complexity of $O(n^2)$ in some cases. However, they insist the time complexity must remain $O(nlogn)$ in all cases. So you proposed a modification to your algorithm for question (a) without increasing the time complexity.

CO1 i. **Name** the algorithm you suggested them and explain the case where the algorithm might take $O(n^2)$ time. **03**

CO2 ii. What solution did you come up with to ensure $O(nlogn)$? **Explain** where it differs from the solution of question (a). **02**

3 A football team's analytics department hired you and gave you a task of reviewing match performance data from the past season. Each match's net goal difference (goals scored minus goals conceded) is recorded in a list. Now you were asked by the department to identify the team's best consecutive performance streak, where the net goal difference was maximized over a series of matches.

net_goal_difference: [1, 4, 3, -5 , 5, 6, 1, -4]

To prove your skills, you used an approach where the problem is divided into subproblems to find the solution. Now tell us more about it by answering the following questions.

CO1 i. For the given input above, **determine** the number of times the *Cross-Sum* exceeds both the *Left-Sum* and the *Right-Sum*. Exclude base cases (subarrays of size 1) from consideration. Show your work properly. **06**
Cross-Sum: sum of elements calculated by combining sum from mid to left and mid to right of the current subarray.
Left-Sum: maximum sum in the left half of the subarray.
Right-Sum: maximum sum in the right half of the subarray.

CO3 ii. The department wants you to find the maximum consecutive matches with a positive goal difference. Now **present** your idea to solve the problem considering you have to use the same approach used earlier with pseudocode/programmable code/step-by-step logical instructions. **04**

B

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Semester Exam | Semester: Fall 2024
Duration: 1 Hour 15 Minutes | Full Marks: 30

CSE 221: Algorithms

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

Name:	ID:	Section:

1 **a.** **CO2** Consider the following code snippet to answer questions (i) and (ii).

```
1.  Function F(n):
2.      If (n == 1):
3.          Return True;
4.      Else:
5.          U = F(n/3);
6.          X = F(n/3);
7.          Y = F(n/3);
8.          Z = F(n/3);
9.          V = A(n);
10.
11. Function A(n):
12.     Sum = 0
13.     For i := 1 to n; i++:
14.         Sum = Sum + i;
15.     Return sum;
```

i. **Write** the time complexity of the function "F" using a recurrence relation. **02**
ii. **Express** the recurrence relation from question (i) with asymptotic upper bound (Big-O) using any convenient method. Show your work. **02**

b. Given an array of **N** elements, you are required to find the **valley element** in it. An element is considered to be a valley if its value is less than the value(s) of its adjacent element(s) i.e. immediate left and immediate right. If there is only one adjacent element, we take only that in consideration. It is guaranteed that the array contains only **one** valley element.

Now answer the following questions.

CO1 i. **Determine** whether each of the following arrays matches the above stem-mentioned conditions: **02**
1. [5, 3, 2, 1, 4]
2. [3, 1, 5, 2, 4]
3. [1, 2, 3, 4, 5]
4. [1, 3, 5, 4, 2]

[N.B.: For each array, it is enough to write only True/False in the answer script.]

CO3 ii. **Present** an algorithm with a time complexity lower than **O(N)** to solve the stem-mentioned problem using pseudocode/ programmable code/ step-by-step logical instructions. **03**

CO2 iii. If the array contains **more than one** peak element, will the presented algorithm in question (ii) still be able to find any peak element? **Explain** briefly. **01**

2 **a.** **CO1** You are designing a computerized logistics system for a warehouse that handles thousands of packages daily. Each package is assigned a unique package ID and a delivery deadline (in terms of 'remaining hours' from the current time) and the packages are initially sorted according to their package_id. The warehouse management system needs to sort the packages by their delivery deadlines in ascending order to prioritize the earliest deliveries. **05**

But there are some conditions to develop this system. The system does not have any issues related to space/memory but it needs the list to be sorted with time complexity $O(nlogn)$ where n is the number of packages to process. Moreover, if any two packages have the same delivery deadline, the package with smaller package_id comes before i.e the algorithm must be stable.

Write the name of the appropriate algorithm for this case and show the solution with pseudocode/programmable code/step-by-step logical instructions.

b. [Continues on the previous question…] After some days, the warehouse noticed a massive increase in daily orders and the system was facing memory issues to hold these massive orders. So they consulted you again. You suggested to them an algorithm which does not require any additional memory. But you also warned them that it can degrade to the time complexity of $O(n^2)$ in some cases. However, they insist the time complexity must remain $O(nlogn)$ in all cases. So you proposed a modification to your algorithm for question (a) without increasing the time complexity.

CO1 i. **Name** the algorithm you suggested them and explain the case where the algorithm might take $O(n^2)$ time. **03**

CO2 ii. What solution did you come up with to ensure $O(nlogn)$? **Explain** where it differs from the solution of question (a). **02**

3 A stock trading firm hired you to analyze daily stock price changes of a company over the past quarter. Each day's net price change (price at the end of the day minus price at the start) is recorded in a list. You were asked by the firm to identify the company's best consecutive stock price increase, where the price change was maximized over a series of days.

Price_changes = [2, 3, 5, -7, 3, 2, 1, -6]

To prove your skills, you used an approach where the problem is divided into subproblems to find the solution. Now tell us more about it by answering the following questions.

CO1 i. For the given input above, **determine** the number of times the *Cross-Sum* exceeds both the *Left-Sum* and the *Right-Sum*. Exclude base cases (subarrays of size 1) from consideration. Show your work properly. **06**
Cross-Sum: sum of elements calculated by combining sum from mid to left and mid to right of the current subarray.
Left-Sum: maximum sum in the left half of the subarray.
Right-Sum: maximum sum in the right half of the subarray.

CO3 ii. The firm wants you to find the maximum consecutive days with a positive price change. Now **present** your idea to solve the problem considering you have to use the same approach used earlier with pseudocode/programmable code/step-by-step logical instructions. **04**