University of Petra		
Faculty of Information Technology		كلية تكنولوجيا المعلومات
Department of Computer Science		قسم علم الحاسوب

Advanced Algorithms
601326
Midterm Exam – 2025 1

Your Name: …….. **Your ID: ……**

Your Instructor Name: ……

Instructions for the Exam:

- Write your name and ID number on the exam and answer sheets.
- Write the number of the section that you enrolled in.
- Write the name of your instructor.
- Questions in the exam not allowed.
- Using any type of technology (mobiles, smart watches) not allowed
- Using extra papers or sheets not allowed
- The exam consists of Six questions.

For instructor use only:

Question number	Course ILO	Program ILO	Question weight	Student mark
Q1			4	
Q2	K2		4	
Q3	I1		8	
Q4			5	
Q5			5	
Q6			4	
Total /30				

Q1) Discuss median-of-three pivot selection and how it enhances Quick Sort algorithm worst case. **(4 marks)**

Median-of-three is a pivot selection strategy in Quick Sort where instead of choosing a single element (like the first, last, or a random element), we:

- **Select three elements from the array, typically the first, middle, and last elements**
- **Find the median value among these three**
- **Use this median as the pivot**

Enhancement of Worst Case:

The standard Quick Sort has a worst-case time complexity of $O(n^2)$, which occurs when the pivot is consistently the smallest or largest element with already sorted or reverse-sorted arrays

Median-of-three tends to produce more balanced partitions leading to O(nlogn).

Q2) Consider the following algorithm. **(4 marks)**

```
Algorithm XYZ (A [0..n − 1])
//Input: An array A[0..n − 1] of n real numbers
val ← 10
res1 ← 0
res2 ← 1
for i ← 0 to n − 1 do
    if A[i] >= val
    res1 ← A[i]+res1
    if A[i] < val/2
    res2 ← A[i]*res2
return res1 – res2
```

a. What does this algorithm compute?
b. What is the time complexity for the algorithm?

Solution:

a. The algorithm returns (sum of elements >= 10) - (product of elements < 5).
b. Total time complexity is O(n).

Q3) **Design** a **divide-and-conquer algorithm** to compute the summation of elements in an array. **(8 marks)**
Setup and solve a recurrence relation for the number of summations made by your algorithm **(use Master Theorem)**

```
function divideAndConquerSum(array A, integer start, integer end)
   if start equals end
      return A[start] // Base case: single element
   else
      mid = (start + end) / 2 // Divide the array into two halves
      leftSum = divideAndConquerSum(A, start, mid) // Recursively compute sum of left half
      rightSum = divideAndConquerSum(A, mid + 1, end) // Recursively compute sum of right half
      return leftSum + rightSum // Combine the sums
```

$S(n)= 2\ S(n/2) +1, \quad$ for $n>1$, $\quad S(1)=0$

Q4) Consider the following recursive algorithm: **(5 marks)**

```
Algorithm Q(n)
//Input: A positive integer n
if n =1
    return 1
else
    return Q(n − 1) + 2 ∗ n – 1
```

a) **Setup** a recurrence relation for the number of multiplications made by this algorithm.
b) **Solve** the recurrence relation using backward substitution and find Big O.

Solution:

a. $M(n)= 0$ for $n=1$, $M(n)= M(n-1) + 1$

b. $M(n)=M(n-1)+1$
$M(n-1)=M(n-2)+1$
$M(n-2)=M(n-3)+1$
$M(n)=[M(n-3)+1]+2=M(n-3)+3$

Using base case, when $n-k=1 \Longrightarrow k=n-1$

$\Rightarrow M(n)=M(1)+(n-1)$

$=0+(n-1)$

$M(n)= n-1$

O(n)

Q5) Given the following array, A= -3 ,8, 4, 22, 13, 6, 20, 9
Apply Quick Sort algorithm to sort the array elements in ascending order **(show detailed steps)** **(5 marks)**

Solution:

a.

Step 1: Initial Call (Full Array)

- **Pivot = -3 (first element)**
- **Since all elements are > -3, the partition is:**
  - **Left subarray: []**
  - **Right subarray: [8, 4, 22, 13, 6, 20, 9]**
- **Result: [-3] + sorted([8, 4, 22, 13, 6, 20, 9])**

Step 2: Sort Right Subarray [8, 4, 22, 13, 6, 20, 9]

- **Pivot = 8**
- **Partitioning:**
  - **Left (≤ 8): [4, 6]**
  - **Right (> 8): [22, 13, 20, 9]**
- **Result: [4, 6] + [8] + sorted([22, 13, 20, 9])**

Step 3: Sort [4, 6]

- **Pivot = 4**
- **Partitioning:**
  - **Left (≤ 4): []**
  - **Right (> 4): [6]**
- **Result: [4] + [6] (already sorted)**

Step 4: Sort [22, 13, 20, 9]

- **Pivot = 22**
- **Partitioning:**
  - **Left (≤ 22): [13, 20, 9]**
  - **Right (> 22): []**
- **Result: sorted([13, 20, 9]) + [22]**

Step 5: Sort [13, 20, 9]

- **Pivot = 13**
- **Partitioning:**
  - **Left (≤ 13): [9]**
  - **Right (> 13): [20]**
- **Result: [9] + [13] + [20]**

Final Sorted Array:
[-3, 4, 6, 8, 9, 13, 20, 22]

Q6) Based on your understanding of the covered algorithms, answer the following: (4 marks)

a) **Compare** Selection Sort to Sequential Search algorithms in terms of best and worst cases.
b) **Discuss** briefly the Best Case of Insertion Sort algorithm.

a.

	Selection sort	**Sequential search**
Best case	**n^2**	**1**
Worst case	**n^2**	**n**

b.

The best case for Insertion Sort occurs when the input array is already sorted in ascending order.
Time Complexity: O(n)
Why This Is the Best Case:
In Insertion Sort, we iterate through the array from the second element to the last, and for each element, we compare it with elements in the already-sorted portion to find its correct position.
When the array is already sorted:

- **Each element is already in its correct position**
- **For each element at position i, we make exactly one comparison with the element at position i-1**
- **Since the current element is already greater than or equal to the previous element, no shifting is needed**
- **We immediately move to the next element**