National University of Computer and Emerging Sciences

Lahore Campus

D & A of Algorithms (CS2009)

Sessional-II Exam

Total Time (Hrs): 1
Total Marks: 15
Total Questions: 3

Date: Nov 3rd 2025

Course Instructor(s)

Dr. MB, Dr. SK, Dr. MB, Dr. MAQ, AA, UH, SK

Roll No _____ Section _____ Student Signature _____

Instructions: Answer in the space provided. Do not attach rough sheets with this exam.

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

2 + 1.5 + 6 + 2 = 11.5

5 + 4 + 3 + 2 + 1

Question 1: MCQs [2 Marks]

i. Consider Fibonacci computed via recursion. For n = 5, how many calls are made (without memoization)?
(a) 9 (b) 15 (c) 25 (d) 32

ii. Q2. Consider Fibonacci computed using dynamic programming (top-down approach via recursion). For n = 5, how many calls are made (with memoization)?
(a) 9 (b) 15 (c) 25 (d) 32

Question 2: [3 Marks]

Example:

Items:	1	2	3	4
Cost:	1	2	3	4
Benefit:	10	15	40	50

Knapsack Capacity: 5

1.5

Perform a dry run of the binary knapsack algorithm using dynamic programming for the example above. Fill in the following array with the values computed for all subproblems.

Weight\item	1 (1,10)	2 (2,15)	3 (3,40)	4 (4,50)
1	10	0	0	0
2	10 15	15	50 0	0
3	10	25 40 30	4	0
4	10	25 50		50
5	10	25		60

Question 3: [2.5+3+0.5 = 6 Marks]

You are climbing a staircase with N steps. You can climb either 1 step or 2 steps at a time.
Write a dynamic programming algorithm to find the total number of distinct ways to reach the top.

Example: Input: N = 4, Output: 5

Possible sequences of steps are: (1,1,1,1), (1,1,2), (1,2,1), (2,1,1), (2,2). So, a total of 5 ways.

a) Write the recurrence relation for this problem.

b) Write pseudocode for your DP algorithm.

c) State the time and space complexity.

dp[0...N) // initialization → for(int i = 0 to n) dp[i] = 1 // initialization

dp[0] = 1

(b)
```
StairStep (int N, int dp[])
{ for i = N-1 to 0
  { if (i ≤ 1)
      dp[i] = 1
    else dp[i] = dp[i-1] + dp[i-2] }
  return dp[0]
}
```

6

(a)
```
{ if (N ≤ 1)  1,
  else StairStep(N-1) + StairStep(N-2)
```

(c) $O(2^{n-1}) \approx O(N)$

Space complexity 1

National University of Computer and Emerging Sciences

Lahore Campus

Question 4: [4 Marks]

A company needs to schedule a set of n jobs on a single machine. Each job i has a deadline d[i] and a profit p[i]. Only one job can be scheduled at a time, and each job takes exactly one unit of time. If a job is completed after its deadline, its profit becomes zero. Formulate a greedy strategy to select and schedule the jobs so that the total profit is maximized.

The following greedy strategy, "Sort all jobs in order of decreasing profit, and for each job, assign it to the earliest available slot (starting from slot 1) such that the job's deadline is not violated," does not always produce an optimal solution for the job scheduling problem. The pseudocode of this approach is given below. It fails on the following counterexample.

pseudocode

```
Job = (id, deadline, profit)
schedule(jobs):
   Sort jobs by profit in descending order
   slot = array[1..D]
   total = 0; t = 1
   for job in jobs:
      if job.deadline ≥ t
         slot[t] = job.id
         total += job.profit
         t = t +1
   return slot, total
```

Counter Example:

Job	Deadline	Profit
A	1	25
B	1	30
C	3	35
D	2	20

Optimal solution: selected jobs = B, C, D. Total profit = 85

Solution by the above incorrect greedy algorithm: selected jobs = C, D. Profit = 55

Propose a correct greedy algorithm for the above problem. Clearly explain the greedy choice property used by your algorithm. Write the pseudocode for your algorithm and analyze its time complexity.

Sort the jobs according to their $\frac{\text{Profit}}{\text{Deadline}}$ ratio and select them if they are within deadline constraint