Roll Number:_____ Number of Pages: 02

Thapar Institute of Engineering and Technology, Patiala

Department of Computer Science and Engineering

MID SEMESTER EXAMINATION

B. E. (Second Year): Sem-II (2021/22) Course Code: UCS415

Course Name: Design and Analysis of Algorithms

April 09, 2022 Saturday, 11:00 Hrs – 13:00 Hrs Time: 2 Hours, M. Marks: 35

Name of Faculty: Rajiv Kumar, Maninder Kaur, Shreelekha Pandey, Rajesh Mehta, Mamta Dabra, Yashwant Singh Patel, Vaibhav Pandey, Shruti Aggarwal

Note: Attempt any 5 questions. Attempt subparts of a question in sequence at one place. Assume missing data, if any, suitably.

Q1. (a) Find the recurrence relation and solve it for the function given in **Fig. 1**. **(5)**

(b) Discuss the time complexity of the Kahn's algorithm for topological sorting? **(2)**

```
1.  int test(int n)
2.  { if (n == 0)
3.      return 1;
4.    for (int i = 1, i <= n; i++)
5.      PRINT n;
6.    return test(n/2) * test(2*n/5);
7.  }
```

Fig. 1

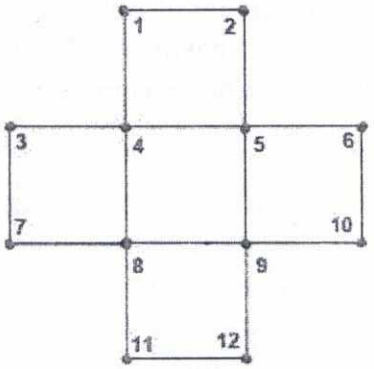

Fig. 2

Q2. (a) Execute Hierholzer's algorithm on the graph shown in **Fig. 2** explaining each step of the algorithm clearly. **(3)**

(b) A project is divided into 10 inter-dependent modules as is shown in the template (**Fig. 3**). The system executes a module only once and stores the result that will be used during later function calls. Determine the possible sequence in which modules get executed by the system to complete the project, giving the required explanations. **(4)**

Module Q { ... ... }	Module R { ... Module Q Module S }	Module S { ... Module Q ... }	Module T { ... Module R Module Q }	Module U { ... Module S Module Z }	Module V { ... Module U Module R }
Module W { ... Module U Module V }	Module X { ... Module W Module T }	Module Y { ... Module W ... }	Module Z { ... Module Y Module V }	Main Module { A = Module R + Module T B = Module Z + Module Y C = Module V + Module X ... }	

Fig. 3

Q3. Use dynamic programming to fully parenthesize the product of five matrices, i.e. $\boldsymbol{A}\,[2 \times 5], \boldsymbol{B}\,[5 \times 3], \boldsymbol{C}\,[3 \times 6], \boldsymbol{D}\,[6 \times 10], \boldsymbol{E}\,[10 \times 7]$ such that the number of scalar multiplications gets minimized. Show each and every step. **(7)**

Q4. (a) Explain how greedy algorithmic strategy differs from dynamic programming by giving at least two distinctions. (2)

(b) Discuss the recursive definition/equation employed in dynamic programming solution for 0/1 knapsack problem. Utilize it to maximize the profit for a knapsack having a capacity of **W = 5** using four items. The respective weight and profit values of the four items are **w[]: (2, 3, 4, 5)** and **p[]: (3, 4, 5, 6)**. (5)

Q5. (a) Five athletes are practicing on a single lane track of **500 meters**. The first athlete starts immediately and completes in **4 minutes**. Second and third athletes start after **5 minutes** taking **12 and 13 minutes** respectively. The fourth one starts **2 minutes** after the second athlete completes, and finishes in **5 minutes**. The last athlete starts **20 minutes** late and completes in **7 minutes**. Determine all the possible schedules that allow the maximum number of athletes to practice? (4)

(b) Mathematically formulate the coin change problem to find the minimum number of coins that add up to the given amount of money. (3)

Q6. (a) Given a sorted array **A** of n distinct positive and negative integers, propose an efficient algorithm to find an index **i** such that $1 \leq i \leq n$ and **A[i] = i**, provided such an index exists. If there are many such indices, then the algorithm can return anyone of them. What is the time complexity of the proposed algorithm? (5)

(b) Can comparison based sorting algorithm be made stable **(Yes/No)**? If **YES**, then explain how the running times of both the versions, stable and unstable, are related? If **NO**, then explain why? (2)

Q7. (a) Discuss possible improvements, with respect to space complexity and time complexity, in the Longest Common Subsequence algorithm using dynamic programming approach. (2)

(b) Write an efficient algorithm or pseudocode for the given scenario. (5)

An **N×N** chessboard with **M** marked squares is there. The row and column numbers of all the marked squares are stored in an array **markedSquares[M][2]**. The task is to visit each of the marked squares exactly once following the rules mentioned below.

- Any of the marked squares can be chosen as the start point.
- Each next visit must be to a marked square in the same row or the same column.
- The directions of the moves must alternate, i.e. one cannot visit in the same row or in the same column twice in any two consecutive moves.

NOTE: It is guaranteed that at least one such visit exists.

----------**ALL THE BEST**----------