Registration Number :

VIT

Vellore Institute of Technology

CHENNAI

Continuous Assessment Test I - February 2024

Programme	B.Tech.(CSE)	Semester	Winter 2023-24
Course	Design and Analysis of Algorithms	Code	BCSE 204L
Faculty	Dr B Jayaram, Dr L K Pavithra, Dr M Janaki Meena, Dr.L.Jeganathan	Slot/Class No.	A1+TA1/ CH2023240502391/ CH2023240502392/ CH2023240502393/ CH2023240502395
Time	90 Minutes	Max. Marks	50

Instructions:

- Answer all the FIVE questions.
- If any assumptions are required, assume the same and mention those assumptions in the answer script.
- Use of intelligence is highly appreciated.
- Your answer for all the questions should have both the 'design' component and the 'analysis component'
- The 'Design' component should consist: understanding of the problem, logic to develop the pseudocode, illustration, pseudocode.
- The 'Analysis' component should consist: Proof-of-Correctness, Computation of T(n), Time-complexity.

---

1. Consider an array A of positive integers with size n. Let the starting index of A be 1 and the ending index of A be n. A Special Sub-array (SSa) of an Array of size n is defined as a sub-array whose starting index is either 1 or the ending index of the subarray is n. For example, The SSa's of $A[1, 7, 6, 2, 3, 4]$ are : $[1], [1, 7], [1, 7, 6], [1, 7, 6, 2], [1, 7, 6, 2, 3], [1, 7, 6, 2, 3, 4], [4], [3, 4], [2, 3, 4][6, 2, 3, 4], [7, 6, 2, 3, 4]$. An SSa is said to be an increasing SSA (i-SSA) if the elements of the subarray are in an increasing order. In the above example, the i-SSa's are : [1], [1,7], [4], [3,4], [2,3,4]. Design an Algorithm to compute all the i-SSa's of the given array A of distinct positive integers, using the 'Divide-Conquer-Combine' strategy. Your algorithm should clearly highlight the 'divide' component, 'conquer' component and the 'combine component', with appropriate comment statements, wherever required. Your design component should contain all the required components. Analyse the algorithm with all the required steps. [10] .

   [**Rubrics**: Logic for pseudocode : 2 marks, Illustration for pseudocode : 3 marks, Pseudocode : 3 marks, Time-complexity :2 marks]

2. Consider an array of size n with positive integres. Let two positive integers a_i, a_j, $a_i \neq 0, a_j \neq 0$, be any two elements of A. We say that a_i is less than or equal to a_j, with respect to the Array A (denoted by $a \leq_A b$) iff $|a_{(i-1)} + a_i + a_{(i+1)})|$ is less than or equal to $|a_{(j-1)} + a_j + a_{(j+1)}|$. That is,

$$a_i \leq_A a_j \quad iff \quad |a_{(i-1)} + a_i + a_{i+1})| \leq |a_{(j-1)} + a_j + a_{(j+1)}|.$$

Note thet the operation $|.|$ is the usual absolute value operation and the operation $\leq$ is the usual less than or equal to operation among the integres. Also note that, a_i is the element in $i^{(th)}$ index of the array A. Similarly, a_j is the element in the $j^{(th)}$ index of the array A. Further, a_k will be zero if $k \leq 0$ or $k > (n+1)$. where n is the size of the array A. Given an array of n positive integers, $a_1, a_2, ..., a_n$, design a pseudocode which will output the positive integers $a'_1, a'_2, ..., a'_n$ such that $a'_1 \leq_A a'_2 \leq_A \cdots \leq_A a'_n$, where the relation '$\leq_A$' is the new relation defined above and $a'_i \in \{a_1, a_2, a_3 ..., a_n\}$, $for\ all\ i$. Your 'design' should involve all the required steps. Analyse your algorithm with all the steps involved. You can follow any strategy for designing the algortihm. [10]

[**Rubrics**: Logic for pseudocode: 2 marks, Illustration for pseudocode : 3 marks, Pseudocode : 3 marks, Time-complexitie :2 marks]

3. Given a positive integer n and the value of e^1 is approximately given as 2.72, Design an Algorithm to compute e^n, where n is a positive integer, using 'Divide-Conquer-Combine' strategy. Here e is the usual exponential operator. Your algorithm should clearly highlight the 'divide' component, 'conquer' component and the 'combine component', with appropriate comment statements, wherever required. Your design component should contain all the required components. Analyse the algorithm with all the required steps. [10] .

   [**Rubrics**: Logic for pseudocode: 2 marks, Illustration for pseudocode : 3 marks, Pseudocode : 3 marks, Time-complexity :2 marks]

4. Propose a problem P in detail (of your choice) **which is not discussed in the classroom or in the lab sessions or in any of the the PPS or in any of the LPS**. Design two different pseudocodes A_1, A_2 with two different logic, for the problem P. Compute the time-complexity of both the pseudocodes A_1 and A_2.
   [10]

   [**Rubrics**: Problem Proposal : 2 marks, logic for A_1 and A_2 : 2, Illustration for A_1 and A_2 : 2 marks, Pseudocodes A_1 and A_2 : 2 marks, Time-complexity of A_1 and A_2 :2 marks]

5. Consider the following algorithm.

Algorithm 1 XXXX(A, w,h)

```
Input : (A,w,h)
if w=h then
    Return 0
end if
c=0
m=⌊(w + h)/2⌋
m1=⌊((m + 1) + h)/2⌋
c +=XXXX(A,w,m)
c +=XXXX(A,m+1, m1)
if h-w > 2 then
    c += XXXX(A, m1 + 1, h)
end if
c += YYYY(A,w,m, m1)
c += YYYY (A,m, m1+1,h )
return c
Algorithm YYYY(A,x,y,z)
c=0
for i= x to y do
    for j= y+1 to z do
        if A[i] = A[j] then
            c +=1
        end if
    end for
end for
Return c
```

Understand the above algorithm and answer the following.

(a) Compute the output of the Algorithm XXXX if the input array is [1,2,9,0,-1, -1, 0] [2 Marks]

(b) Describe the functionality of the Algorithm XXXX [2 Marks]

(c) Compute the time-complexity of the Algorithm XXXX [2 Marks]

(d) Modify the Algorithm XXXX into another Algorithm BB such that the functionality of Algorithm XXXX and the functionality of Algorithm BB remain same. [4 Marks]

$\Longleftrightarrow \Longleftrightarrow \Longleftrightarrow$