|  |  |  |
| --- | --- | --- |
|  | **I2206 E**  **Data Structures**  **Dr. Zein Al Abidin IBRAHIM & Dr. Ali Jaber** | **Final 2024 - 2025**  **Monday 23 June 2025**  **Duration 90 minutes** |

**Question I: Complexity [3+3+4+4+6=20 pts, 15 minutes]**

Find and prove the complexity of each of the following:

1. What is the complexity of the following piece of code?

void mysteryFunction1(int n) {

int i = 1;

while (i < n) {

printf("%d ", i);

i \*= 2;

}

}

|  |
| --- |
|  |

1. What is the complexity of the following piece of code?

void mysteryFunction2(int n) {

for (int i = 0; i < n; i++) {

for (int j = i; j < n; j++) {

printf("%d %d\n", i, j);

}

}

}

|  |
| --- |
|  |

1. Given the following recurrence relation that we want to derive the complexity of such problem.

|  |
| --- |
|  |

1. Given the following recurrence relation that we want to derive the complexity of such problem:

|  |
| --- |
|  |

1. What is the complexity of the following piece of code?

void mysteryFunction5(int n) {

int count = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < i \* i; j++) {

if (j % 2 == 0) {

for (int k = 0; k < j; k++) {

count++;

}

}

}

}

}

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Formulas:**  **Geometric Series**  **Harmonic Series**  **Other important formulas** | **Master Theorem for Analysis of Algorithm - Krantesh SinghMaster Theorem** |

**Question II: Algorithms [10+20+30+20=80 pts, 10+15+30+25=75 minutes]**

1. Given a sorted array of integers that represents elements of arithmetic progression in order. Some elements may be missing in the progression. Write a function in c that takes an array as input with its size and prints all the missing numbers. Suppose here that we have the same progression, but some values are missing.

**Examples:**

***Input****: arr[ ] = [2, 4, 8, 10, 12, 14]*

***Output****: 6*

***Explanation****: In this case, the common difference is 2, and 6 is missing between 4 and 8.*

***Input****: arr[ ] = [1, 11, 16, 21, 36, 41]*

***Output****: 6, 26,31*

***Explanation****: In this case, the common difference is 5, and 6, 26 and 31 are missing.*

**Attention** that the difference between the first two values may not be the value of the progression since the second value may be a missing one. Just take the minimal difference between all two consecutive values.

|  |
| --- |
|  |

1. Given a **Binary search tree,**the task is to write a recursive function to **delete all** the**leaf nodes** from the binary search tree.

|  |  |
| --- | --- |
| 🡪 | typedef struct BTNode{  element data;  struct BTNode \*left, \*right;  } BTNode; |

|  |
| --- |
|  |

1. You are working with a **Single Threaded Binary Search Tree** (**STBST**). In this specific type of BST, all NULL right child pointers are "threaded" to point to the in-order successor of the node. An integer flag, rightIsThread, within each node distinguishes between a regular right child pointer (0) and a thread (1). Left child pointers behave as in standard BST (pointing to a child or NULL). Here is the structure definition:

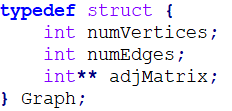
|  |  |
| --- | --- |
| In the right figure, the nodes 1, 5, 7, 9 have the rightIsThread value = 1 and their right pointer points to the successor node in the in-order traversal while nodes 3, 6, 8, 11, 13 the rightIsThread is 0. | threadedBT |

1. Implement a C function inorder\_traverse\_single\_threaded\_bst(Node\* root) that performs an iterative in-order traversal of a given Single Threaded Binary Search Tree. You are not allowed here to do a recursive function nor using any auxiliary data structure like array, stack, queue, etc. Think to use the rightIsThread flags and the threads to move through the tree.

|  |
| --- |
|  |

1. Implement a C function insert\_into\_single\_threaded\_bst(Node\* root, int value) that inserts a new value into a Single Threaded Binary Search Tree while maintaining both the BST property and the single-threading property. If the tree is initially empty, the function should return the new root. Otherwise, it should return the original root. In case the value already exists, just ignore the insertion.

|  |
| --- |
|  |

1. You are given a positive weighted directed graph. This graph is represented by the right Graph structure which contains the number of vertices, the number of edges, and its adjacency matrix. For each node in the graph, you need to calculate the absolute difference between the total sum of its indegree weights and the total sum of its outdegree weights.

Your task is to implement a C function named calculateDegreeWeightDifferences that takes a pointer to a Graph structure as input. The function should dynamically allocate an integer array, populate it such that result[i] contains the absolute difference for node i, and then return a pointer to this newly allocated array.

**Assumptions:**

* The graph nodes are indexed from 0 to N-1, where N is the number of vertices.
* The adjacency matrix adjMatrix[i][j] represents the weight of the directed edge from node i to node j.
* If there is no edge from i to j, adjMatrix[i][j] will be 0.
* All weights are positive integers.
* Your solution should not exceed O(N2)

|  |  |
| --- | --- |
| **Example:**  Node 0: has a total in-degree weights = 4 and total out-degree weights = 4+3 = 7 so the absolute difference is 7-4 =3.  Node 1: (4+4) – (5 + 2) = 1  Node 2: (5+3) – 7 = 1  Node 3: (7+2) – 2 = 7  Node 4: 2-(6+4+4) = 12  Node 5: 6-0=6 |  |

|  |
| --- |
|  |

Scratch

Scratch

Scratch

Scratch