

Foundation Exam Structure (January 2022 – beyond)

Section A: Basic Data Structures

1. Dynamic Memory Management in C - Tracing/Coding

- i. Dynamically allocating memory for a struct**
- ii. Dynamically allocating memory for an array**
- iii. Dynamically allocating memory for a 2D array**
- iv. Dynamically allocating memory for an array of arrays.**
- v. Solving problems with arrays.**
- vi. Freeing memory in all cases**

2. Linked Lists – Tracing/Coding

- i. How to allocate space for a new node (malloc)**
- ii. When to check for NULL**
- iii. What free does**
- iv. Iteration vs. Recursion**
- v. Insertion**
- vi. Deletion**
- vii. Structural Modification**

3. Abstract Data Structures – Tracing/Coding

- i. Stacks**
 - a. Converting infix to postfix expressions**
 - b. Evaluating postfix expressions**
 - c. Array Implementation**
 - d. Linked List Implementation**
- ii. Queues**
 - a. Array Implementation**
 - b. Linked List Implementation**

Section B: Advanced Data Structures

1. Binary Trees – Tracing/Coding

- i. How to allocate space for a new node (malloc)**
- ii. When to check for NULL**
- ii. Tree Traversals**
- iii. What free does**
- iv. Using recursion with trees**
- v. Computing sum of nodes**
- vi. Computing height**
- vii. Other variants**

2. Advanced Data Structures - Tracing/Coding

- i. Hash Tables**
 - a. Hash Function Properties**
 - b. Linear Probing Strategy**
 - c. Quadratic Probing Strategy**
 - d. Separate Chaining Hashing**
- ii. Binary Heaps**
 - a. Insertion**
 - b. Delete Min/Max**

3. Advanced Tree Structures

- i. AVL Trees**
 - a. Tracing inserts**
 - b. Tracing deletes**
 - c. Searching for a value**
- ii. Tries**
 - a. Tracing inserts**
 - b. Searching for a word**

Section C: Algorithm Analysis

1. Algorithm Analysis

- i. Known Data Structures**
- ii. Best, Average, Worst Cases**
- iii. Based on various implementations**
- iv. New Problem Analysis**

2. Timing questions

- i. Set up correctly with an unknown constant**
- ii. Solve for the constant.**
- iii. Use direct formula to answer the question**
- iv. For loop questions, write out summations**

3. Summations and Recurrence Relations

- i. Break them down into multiple summations if necessary**
- ii. Evaluate each of those using summation formulas.**
- iii. Remember that indices of summation are important.**
- iv. The n in the formula is JUST a variable!!!**
- v. Deriving recurrence relation from code**
- vi. Using iteration to solve recurrence relations**

Section D: Algorithms

1. Recursive Coding

- i. Need a terminating condition**
- ii. Need an algorithm for non-terminating case.**
- iii. In particular, you must reduce a question to “smaller” instances of the same question.**
- iv. Do not try to think of an iterative solution!!!**
- v. Towers of Hanoi solution and recursion**
- vi. Permutation**
- vii. Floodfill**

2. Sorting

- i. Insertion Sort**
- ii. Selection Sort**
- iii. Bubble Sort**
- iv. Merge Sort (Merge)**
- v. Quick Sort (Partition)**

3. Base Conversion and Bitwise Operators

- i. Base Conversion**
 - a. Converting from base b to base 10.**
 - b. Converting from base 10 to base b.**
 - c. Converting between bases 2, 4, 8, 16 through base 2.**
- ii. Bitwise Operators to express subsets**
 - a. Mechanics of &, |, ^, >>, <<.**
 - b. Corresponding "set" meanings.**
 - c. How to check if a bit is "on" or "off" in a number.**