PRACTICE EXAM

Difficulty: MEDIUM

Questions: 10

Data Structures Exam - Medium Difficulty

Instructions: Answer all questions to the best of your ability. Show your work where applicable.

Section 1: Multiple Choice (4 points each, 40 points total)

Instructions: Choose the best answer for each question.

Question 1: Which sorting algorithm performs best in the worst-case scenario, exhibiting a time complexity of O(n log n)?
A) Insertion Sort
B) Heap Sort
C) Bubble Sort
D) Selection Sort

Question 2: In a heap data structure, which of the following represents the index of the right child of a node A[i]?
A) A[i/2]
B) A[i-1]
C) A[2i]
D) A[2i + 1]

Question 3: What condition signifies a "stack overflow"?
A) Attempting to pop an element from an empty stack.
B) Attempting to push an element onto a full stack.
C) Attempting to dequeue an element from an empty queue.
D) Attempting to enqueue an element into a full queue.

Question 4: In a linked list, what does `prev[x] = NULL` signify?
A) The element x is the tail of the list.
B) The element x is the head of the list.
C) The element x is located in the middle of the list.
D) The list is empty.

Section 2: Short Answer (6 points each, 30 points total)

Instructions: Answer each question in 2-3 sentences.

Question 5: Briefly explain the difference between a queue overflow and a queue underflow.

Question 6: Explain how a linked list differs from an array in terms of memory allocation and the ordering of elements.

Question 7: Describe the purpose of the LIST-SEARCH(L, k) procedure for a linked list.

Section 3: Problem-Solving (10 points each, 30 points total)

Instructions: Provide detailed solutions for each problem.

Question 8: Illustrate the first two iterations (j=2, then j=3) of the Insertion Sort algorithm on the following array: `[5, 1, 4, 2, 8]`. Show the state of the array after each iteration.

Question 9: Given a heap represented by the array `[20, 12, 13, 11, 7, 9, 2, 1]`, draw the corresponding binary tree representation of the heap.

Question 10: Write pseudo-code to describe how you would merge two sorted arrays into one. Assume that you are merging `array1` and `array2` into `mergedArray`.