C Programming Practice Questions

1. Pointers

1. Write a C program to demonstrate pointer declaration and initialization.

2. Write a function to swap two numbers using pointers.

3. Write a program to find the length of a string using pointers.

4. Implement a program to reverse a string using pointers.

5. Write a program to access and modify array elements using pointers.

6. Write a program to compare two strings using pointer arithmetic.

7. Implement pointer to pointer (double pointer) example to dynamically allocate memory for a 2D array.

8. Write a program to count vowels in a string using pointers.

9. Demonstrate use of pointers with structures.

10. Write a C program to dynamically allocate memory for an array using malloc and free.

2. Arrays

1. Write a C program to find the largest element in an array.

2. Implement linear search in an array.

3. Write a program to reverse an array in place.

4. Write a program to merge two sorted arrays.

5. Implement insertion of an element in an array at a given position.

6. Write a C program to remove duplicates from a sorted array.

7. Write a function to rotate an array by k positions.

8. Implement a function to find the second largest element in an array.

9. Count frequency of each element in an array.

10. Check if an array is a palindrome.

3. Stack

1. Implement a stack using arrays.
2. Write push and pop operations for a stack implemented using arrays.
3. Implement a stack using linked list.
4. Write a function to reverse a string using stack.
5. Check for balanced parentheses in an expression using stack.
6. Evaluate a postfix expression using stack.
7. Convert an infix expression to postfix using stack.
8. Implement peek and isEmpty functions for stack.
9. Write a program to implement two stacks in one array.
10. Implement a stack to store and retrieve min element in O(1) time.

4. Queue

1. Implement a queue using arrays.
2. Write enqueue and dequeue functions for a queue implemented using arrays.
3. Implement a queue using linked list.
4. Implement a circular queue.
5. Write a program to implement a priority queue.
6. Implement a queue using two stacks.
7. Implement dequeue (double-ended queue).
8. Check if a given queue of integers is a palindrome.
9. Simulate a real-world queue (e.g., printer job queue).
10. Write a program to reverse a queue using a stack.

5. Linked List

1. Write a program to create a singly linked list.
2. Implement insert at beginning, end and at a given position in singly linked list.

3. Delete a node from beginning, end and a given position in singly linked list.

4. Reverse a singly linked list.

5. Write a function to detect loop in a linked list.

6. Implement a doubly linked list.

7. Write a function to merge two sorted linked lists.

8. Find the middle element of a linked list.

9. Implement a circular linked list.

10. Write a function to remove duplicates from a sorted linked list.

11. Detect and remove a loop in linked list.

6. Searching and Sorting Algorithms

1. Implement linear search.

2. Implement binary search (iterative and recursive).

3. Implement bubble sort.

4. Implement insertion sort.

5. Implement selection sort.

6. Implement merge sort.

7. Implement quick sort.

8. Write a program to find the kth smallest element in an array.

9. Count the number of comparisons in different sorting algorithms.

10. Implement a program to sort strings alphabetically.

11. Write a program to sort an array using heap sort.