Algorithm Lab Report:

Caution: Using ChatGPT or copying assignment code from any external sources will result in an automatic grade of 0. Don't cheat. Try to solve as many problems as you can. Your marks will be based on your efforts.

Part1: Basic Problem

1. Write a program that checks whether a number entered by the user is prime or not.
2. Develop a program to find the factorial of a given number using a loop.
3. Implement a program that converts temperature in Celsius to Fahrenheit and vice versa.
4. Create a program to check if a given string is a palindrome.
5. Write a program that generates the Fibonacci series up to a given number.
6. Develop a program that counts the number of words in a given sentence.
7. Write a program that calculates the area and perimeter of a rectangle given its length and width.

Part2: Recursion

1. **Factorial Calculation:**

   Write a C function to calculate the factorial of a non-negative integer 'n'. The factorial of 'n' is the product of all positive integers from 1 to 'n'.

2. **Fibonacci Series:**

   Create a C function to find the nth number in the Fibonacci series. The Fibonacci series starts with 0 and 1, and each subsequent number is the sum of the two preceding ones.

3. **Sum of Array Elements:**

   Implement a C function that calculates the sum of elements in an integer array using recursion.

4. **Exponentiation:**

   Write a C function to calculate the result of raising a base 'b' to a non-negative integer exponent 'e'.

5. **Binary Search:**

Implement binary search in C using recursion to find an element in a sorted integer array.

6. **Reverse a String:**

   Write a C function to reverse a given string using recursion.

7. **Count Digits in an Integer:**

   Implement a C function that counts the number of digits in a positive integer using recursion.

8. **Sum of Natural Numbers:**

   Create a C function to calculate the sum of the first 'n' natural numbers using recursion.

Part3. Sorting Algorithm

1. **Bubble Sort:** Implement the bubble sort algorithm to sort an array of integers in ascending order.

2. **Bubble Sort Optimized:** Implement an optimized version of the bubble sort algorithm to sort an array of integers in ascending order.

3. **Selection Sort:** Create a program to sort an array of strings using the selection sort algorithm in ascending order.
4. **Insertion Sort Descending Order:** Implement the insertion sort algorithm to sort an array of floating-point numbers in descending order.
5. **Quick Sort:** Implement the Quick sort algorithm to sort an array of integers in ascending order.
6. **Merge sort:** Implement the merge sort algorithm to sort an array of integers in ascending order.

Part4: Linear Search & Binary search

1. **Array Sum:** Given an array of integers, write a program to find the sum of all elements using linear search.

2. **Maximum Element:** Find the maximum element in an array of integers using linear search.

3. **Element Frequency:** Count the frequency of a specific element in an array using linear search.

4. **Unsorted Array Search:** Given an unsorted array, write a program to find a specific element using linear search.
5. **Linear Search on a Linked List:** Implement linear search to find a specific element in a linked list.
6. **Sorted Array Search:** Given a sorted array of integers, write a program to find a specific element using binary search.

7. **First Occurrence:** Given a sorted array with duplicate elements, find the first occurrence of a specific element using binary search.

8. **Square Root Approximation:** Implement a function to find the square root of a given positive number using binary search with a specified precision.

9. **Bitonic Array Maximum:** Given a bitonic array (first increasing, then decreasing), find the maximum element using binary search.

10. **Peak Element:** Find a peak element in an array. An element is considered a peak if it's greater than or equal to its neighbors.

Part5: Greedy and DP Problem

1. **Fibonacci Series:** Implement a program to find the nth Fibonacci number using dynamic programming, storing previously calculated values to avoid redundant computations.

2. **Knapsack Problem:** Solve the 0–1 knapsack problem where you have a set of items with weights and values and need to maximize the value while staying within a given weight limit.

3. **Coin Change Problem:** Determine the number of ways to make a certain amount using a given set of coins.

4. **Matrix Chain Multiplication:** Given a sequence of matrices, find the optimal way to parenthesize them to minimize the number of scalar multiplications.

5. **Longest Increasing Subsequence:** Find the length of the longest increasing subsequence in an array.

6. **Subset Sum Problem:** Determine if there is a subset of given integers that adds up to a specified sum.
7. **Greedy Coin Change Problem:**

   Problem Statement: Given a set of coins with different denominations `c1, c2, ... cn` and a value `V`, find the minimum number of coins needed to make the change for the value `V`. Assume you have an infinite number of each of your coin denominations.

   Example:

   Coins: `1, 2, 5`
   Value: `11`
8. **Greedy Knapsack Problem:**

   Problem Statement: Given weights and values of n items, put these items in a knapsack of capacity `W` to get the maximum total value. Each item can be selected once.

   Example:

   Weights: `10, 20, 30`
   Values: `60, 100, 120`
   Knapsack Capacity: `50`

Bin Packing Algorithm:

Problem Statement: Given a set of items with weights `w1, w2, ... wn` and bins with the same capacity `C`, find the minimum number of bins required to place all items such that no bin overflows.

Example:
Items: `4, 8, 1, 4, 2, 1`

Bin Capacity: 10

Huffman Coding Algorithm Problem:

Problem Statement: Given a string S and its frequency count for each character, design an optimal prefix-free binary code (Huffman code) for the characters with the least average code length.

Example:

String: BACADAEAFABBAAAGAH

9. DP Knapsack Problem:

Given a set of items, each with a weight and a value, determine the maximum value you can obtain by selecting a subset of the items while ensuring their total weight does not exceed a given capacity.

Example:

```
Items:  A  B  C
Values: 6 10 12
Weights: 2  4  6
Knapsack Capacity: 8

Maximum Value: 22
```

10. **DP Coin Change Problem:**

Given a set of coin denominations and a target amount, find the minimum number of coins needed to make up the target amount.

Example:

```
Denominations: 1  5  10
Target Amount:  12
```