Data Structure Practice Problems

Sheet 1 (upto Array based Polynomial, Sparse Matrix, & Searching, Sorting) (Jan 2026)

NOTE:

1. WAP is abbreviation for "Write A Program". WAMP is abbreviation for "Write A Modular Program".

2. Questions/programs marked as * are difficult, programs marked with ** are more difficult. Good programmers/students should try most of these.

3. From topic 5 (stack based programs), all programs must be made modular

Topic 1a: Control statements Based

1. Write a program (WAP) that asks the user to enter a fraction (you can read it in any format as per your convenience), and then convert the fraction to its lowest term. For example, 24 / 36 should get converted to 2 / 3
2. Using for loop, print following pattern of N lines(N will be input from user)

A	1
ABC	121
ABCDE	12321
... N lines (N>=26)	
(After Z start using A,B again)	1234567890987654321
	123456789010987654321
	... N Lines

3. Using loops, find sum of the series:
   $S = x - x^3/3! + x^5/5! - x^7/7! + ... x^N/N!$ (assume N is always odd)

Implement in two ways: (a) computer each term using two user-defined functions: power, fact (b) In the whole program, use only one loop (for-loop).

Topic 1b: String Based:

1. Write a program that accepts as input a string and determines the frequency of occurrences of each of the distinct characters in string. Find (i) which is the first repeating character (ii) the character repeated the most.

2. Write a function, strndel, that accepts a string and two integers, start and length. Return a new string that is equivalent to the original string, except that length characters beginning at start have been removed.
3. Write a function, strdel, that accepts a string and a character. The function returns string with the first occurrence of character removed.
4. Write a function censor(…) that modifies a passed string by replacing all occurrences of the letters "bye" with "new". For example, string "goodbye bye one" would become "goodnew new one". Make the function fully readable and structured. Out of standard string functions, only strlen can be used.
5. Write a program that reverses words of a sentence in a manner given in the following example: "Android is a flavour of Linux" will be reversed to "Linux of flavour a is Android"
6. Write a program to concatenate two strings using strcat
7. WAP to concatenate the second string on the left of the first string without using strcat. For example if first string is "world", second string is "hello", then the first string would become "helloworld". Try it in two ways: (i) you can use extra array (ii) use of any additional array is not permitted.

Topic 2: Array Based:

1. Write a program which reads an array of n elements. Now check whether all elements are in ascending order or not. Don't sort array.
2. Write a program which reads an array of n elements. Find sum1 of elements at index 0, 2, 4, 6… & sum2 as elements of index 1, 3, 5, 7, ….. For computing sum1/sum2, you can use loop only once and "for" loop is to be used. Both sum1 and sum2 should get computed via this single "for" loop.
3. WAP to find sum of diagonals of a square matrix. Find sum for both diagonals.
4. Consider a 2-d matrix, WAP to find the row whose sum is lowest among all rows.
5. Write a program to reverse an array by swapping (without using additional memory).
6. Write a program to identify the missing numbers in a given Array within the range [1...N]. Try it in two ways: (i) assume array is sorted (ii) array is not sorted
7. Given an array of n integers, pass this to a function that generates a new array from this such that all even numbers come first followed by all odd numbers. The newly generated array must be returned back through a parameter to the function.
8. Write a program to find largest of a 2-d matrix
9. Program to find the largest of every row of a 2-d matrix and store it at the end of the respective row. Similarly, find the smallest of every column and store it at the end of the respective column.
10. Write a program to find transpose of a matrix (can be non-square) in 2 ways:
    (i) store the transpose in a new matrix
    (ii) store the transpose in the input matrix itself and don't use any extra array/matrix

Topic 3: Polynomial & Sparse Matrix based:

1. Write a program to evaluate a given polynomial for a given value of x. For example, $2x^5-3x^4+4x^3-5x^2+6$ to be evaluated if x=2; Implement for both representations (degree based, and exp-coeff based).
2. For the exp-coeff representation of polynomial, display the o/p so that the exponents appear truly as superscripts:

(i) $20x^{30}-103x^{20}+5x^{10}+256$

2. It will require display in two lines: first line to display exponents based on exact computation of spaces in between two exponents, and second line below will display coeff and "x" etc.
3. For exp-based representation, instead of struct based, store poly as int poly[50][2] and implement poly-addition
4. For degree based representation, in lecture, we stored data in the array such that index i contains coefficient of x^{n-i} . Now assume that index [i] should store coeff of x^i. Reading of input will be obviously in conventional manner, i.e. x^degree onwards. Write necessary code to add two polynomials now.
5. Write a program to add two sparse matrices. Read & display the input/output as row*col order matrix (including zeros) and internal storage & processing should be based on sparse representation i.e. array of struct {row,col, val},
6. Instead of struct based, store sparse representation as int spars[50[3] and write a program to find transpose, fast-transpose, addition
7. Problems from Horowitz & Sahni book:
   (i) Page 72: Q 1, 5, 6
   (ii) Page 84: Q2, 6

8. WAP to multiply two polynomials using two ways: (a) degree based storage of coefficients in an array (ii) (exp, coeff) pairs stored in array in descending order of exp. For both implementations, you can make and use an addpoly function, which is implemented to add two polynomials.

Topic 4: Searching & Sorting based:

1. Consider the following input test cases:

   12, 2, 16, 30, 8, 28, 4, 10, 20, 16

   5 5 10 15 20 25 30 35 40 40

   40 40 30 25 20 20 15 10 5 5

   1 1 5 6 7 5 8 1 5

There is no need to write the code of any sorting algo. You have to apply the respective sorting algo on above test cases one by one and sort:

(a) Use Selection sort and show all main steps sequentially. Each element being swapped should get demonstrated clearly.

(b) Use Insertion sort and show all main steps. Demonstrate the insertion of each element at its proper index. Whenever an element reaches to its final index, it is a main step.
(c) Use merge sort and show all main steps sequentially. Each time a merge gets completed is a main step. Demonstration of merge process for equal values should be done clearly.
(d) Use quick sort and show all steps sequentially. Demonstrate for each pivot reaching to its correct position and during that show all swaps taking place.

2. Prove that Merge Sort is stable and Quick Sort is unstable. Similarly, find out whether selection and insertion sort are stable or not?
3. On each iteration of its outer loop, insertion sort finds the correct place to insert the next item, relative to the ones that are already in sorted order. It does this by searching back through those items, one at a time. Would insertion sort be speeded up if instead it used binary search to find the correct place to insert the next item?
4. **Implement iterative merge sort.
5. After two passes of sorting algorithm, the following array: 47, 3, 21, 32, 56, 92 has been rearranged as: 3, 21, 47, 32, 56, 92. Which sorting algorithm is being used and why?
6. An array contains the elements: 7, 8, 26, 44, 13, 23, 98, 57. The first two elements have been sorted using Selection Sort. What would be the elements in the array after 4 more passes of the selection sort algorithm?