Question 1 [12+4+6=22 Marks]

Part A. Which of the following is/are stable sorting algorithm(s)? Provide a technical justification for your answer.
*A sorting algorithm is stable if whenever there are two records **R** and **S** with the same key (value) and with **R** appearing before **S** in the original list, **R** will appear before **S** in the sorted list.*
[3 x 4 = 12 Marks]

	Algorithm	Stable/ Unstable	Justification
a)	Merge Sort	Stable	Suppose L[i]=20, R[j]=20, if L[i]<=R[j] then A[k]=L[i], Actual array gets value from left sub-array
b)	Insertion Sort	Stable	Suppose A[i]=20, key=20, if A[i]>key then A[i+1]=A[i] else A[i+1]=key
c)	Quick Sort	Unstable	Elements are swapped considering Pivot instead of their actual positions Example: {4,3,4,1,2}
d)	Counting Sort	Stable	Suppose A[4]=20, A[5]=20, final loop starts with j=length.A to 1, so A[5] will be placed to its position and later positions are decremented.

Part B. Merge Sort does not perform in-place sorting. Explain with a short example *(two-sub arrays with 2 and 3 elements respectively)* how the extra memory is used and why the Space Complexity is O(n).
Note: A complete dry run is not required. (1 + 1 = 2 marks)
In last call of Merge operation two n/2 sized auxiliary arrays are used and previously created local arrays are no more in the memory.
Example:

A=

2	5	1	2	9

L=

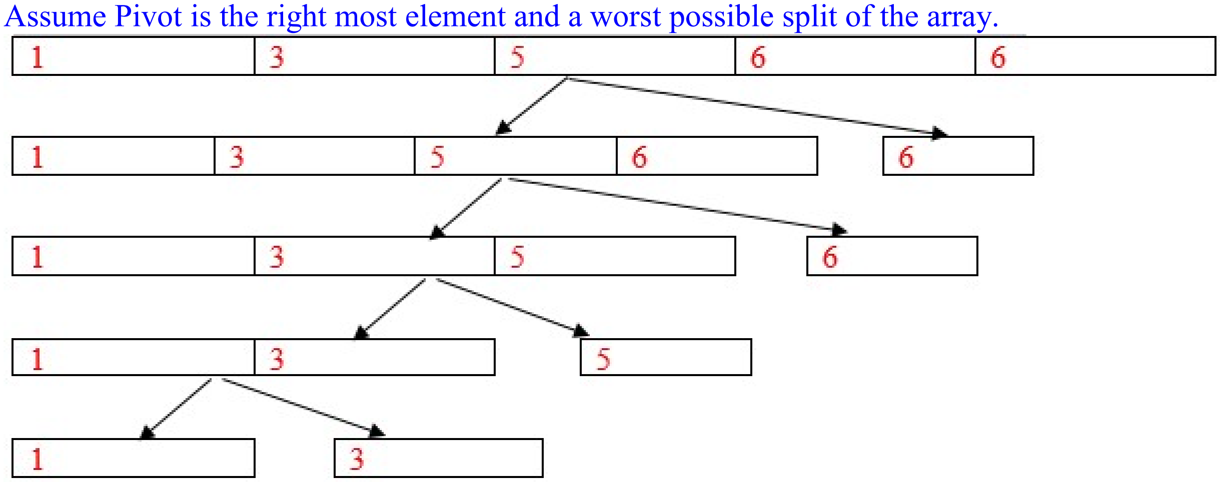
2	5

R=

1	2	9

Part C. Prove with an example consisting of a dry run on an array of five elements that the worst case time complexity of Quick Sort is $O(n^2)$. **[6 Marks]**

Assume Pivot is the right most element and a worst possible split of the array.



Question 2 [5+10=15 Marks]

Consider a 3-ary max-heap. Note that this is different from a binary max-heap. You are required to use the array-based implementation of the heap. ***Start the array index from 1.***

Part A. Provide the formula for accessing the root element. Provide the formulae of accessing the three children of any i-th node. Also, provide the formula for accessing the parent node of any i-th node.

Root of the heap is stored at the array index 1
Children of a node i are stored at the array indices **3i-1** to **3i+1**
Parent of a node i is stored at **floor((i+1)/3)**

Part B. Complete the MAX-HEAPIFY-3-ary procedure given below for the above-mentioned 3-ary max-heap and perform the time complexity analysis. (Note that you are required to modify the MAX-HEAPIFY procedure of binary max-heap)

```
MAX-HEAPIFY-3-ary(A, i, n)
BEGIN
l ← 3×i - 1
m ← 3×i
r ← 3×i + 1
largest = i
IF l ≤ n and A[l] > A[i] THEN
largest ← l
IF m ≤ n and A[m] > A[l] THEN
largest ← m
ELSE largest ← l
IF r ≤ n and A[r] > A[largest] THEN
largest ←r
IF largest ≠ i THEN
exchange A[i] ↔ A[largest]
MAX-HEAPIFY-3-ary(A, largest, n)
ELSE
            return
END
```

Best case = O(1); Worst case = Average case = O (log n)

Question 3 [3+10+2=15 Marks]

The following information is based on a piece of text using a set of five different symbols. The frequencies of the symbols in the text are given below:

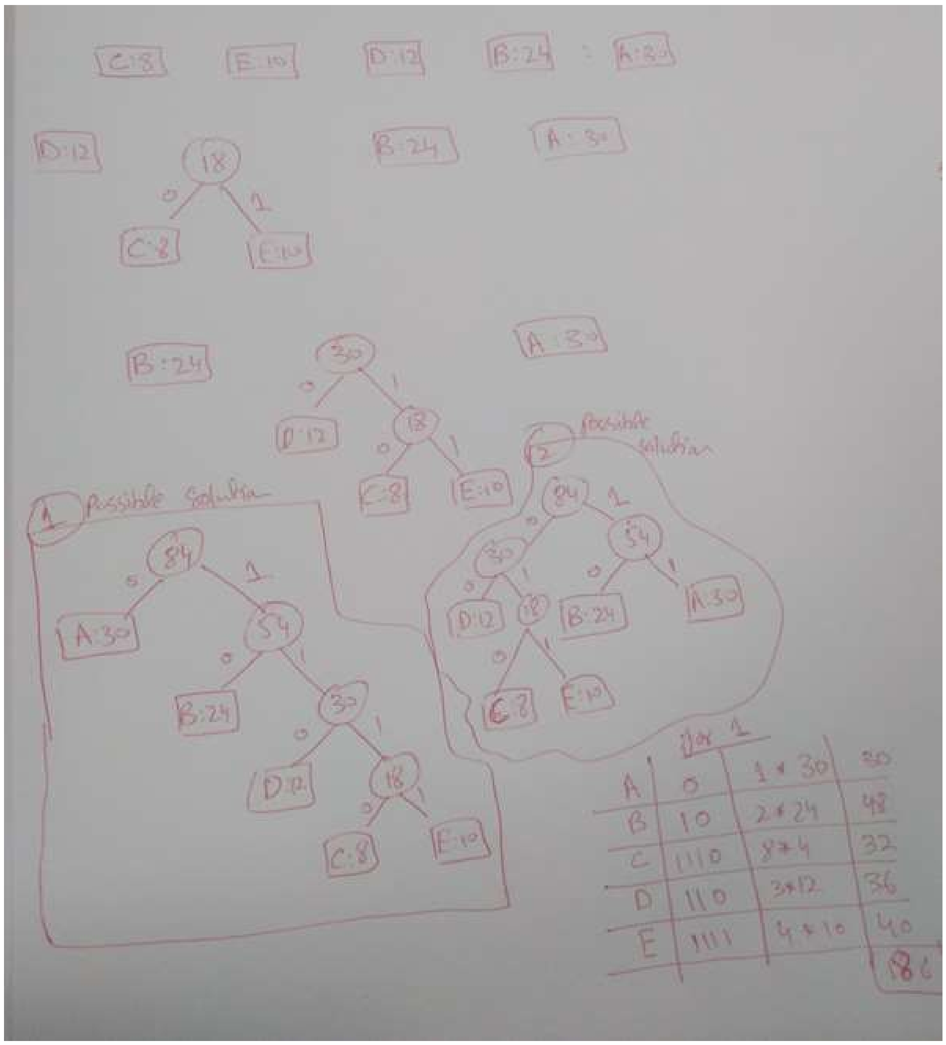
Symbol	Frequency
B	24
D	12
A	30
E	10
C	8

Part A. What is the minimum number of bits required to store the text using a fixed-length coding scheme? Justify your answer. **[3 Marks]**

252 bits are required to store the text
(total characters in the text = 84; min. bits required to store a character = 3)

Part B. What is the minimum number of bits required to store the text using a variable-length coding scheme? You are required to use the Huffman's algorithm learnt in the class. Justify your answer by showing all steps. **[10 Marks]**

Part C. Show the final Huffman's tree. **[2 Marks]**



Question 4 [6+8+6+8=28 Marks]

Part A. The suffix function σ(x) specifies the length of the longest prefix of pattern **P** that is also a suffix of **x**. **[6 Marks]**

Show the result of the following suffix function σ(x), for the following patterns
[*Note:* *Your result should be a number]*

i. Pattern= "university" σ("fastuni")= _____3____

ii. Pattern=" algo" σ("designalgo")= __4______

iii. Pattern= "datastructure" σ("datastructures")= _0_____

Part B. Draw the state transition table and state transition diagram for the pattern **P=aabab** over the alphabet ∑ ={a,b} using the following Compute-Transition-function. Construct the string-matching automaton for the pattern and illustrate its operation on the text string
T=aaababa **[8 Marks]**

NOTE: Zero marks for only showing table and state transition diagram without showing dry run for every step for making transition table

Partial Solution:
0→1→2→2→3→4→5→1→0→1
Complete dry run

Part C. Compute the prefix function ∏ for the pattern **P=babababaaaab** **[6 Marks]**

P[i]	b	a	b	a	b	a	a	a	a	b
∏ [i]	**0**	**0**	**1**	**2**	**3**	**4**	**0**	**0**	**0**	**1**

Part D. Suppose that all characters in the pattern P are different. Show how to accelerate NAIVE-STRING-MATCHER to run in time O(n) on an n-character text T. **[8 Marks]**

```
Accelerated_naive_algo(T,P,n,m)
BEGIN
    s ← 0
    WHILE s ≤ n-m DO
        FOR (j : 1 to m) DO
            IF P[j] ≠ T[s+1] THEN
                break
            END IF

            IF j = m THEN
                PRNT "Pattern occurs at shift : ", s-m
            END IF

            s ← s + 1
        END FOR
    END WHILE
END
```