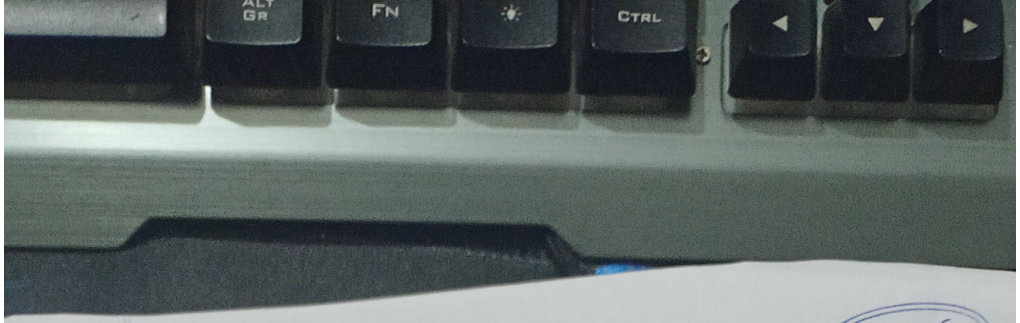


Saathi

Date ___/___/___

Data Structures :-

SET - 1

Quest Define asymptotic notation and explain it importance in analysing algorithm efficien

Ans) Asymptotic notation is a mathematical tool use to describe the growth rate of an algorithm's running time or space requirements.

- It helps to determine the efficiency of an algorithm.

- Common Notation :-

i) Big O (O) :- Worst case
ii) Omega (Ω) :- Best case
iii) Theta (Θ) :- Average

Quest Explain the difference between tail rec and head recursion.

Ans) Tail recursion : The recursion call is t last statement in the function.

Ex → int count (int n) {
if (n == 0) retu
print n;
count(n-1); ->
}

Saathi

Head recursion : The recursive call is the first statement after the termination condition

Ex → void count (int n) {
if (n == 0) return
count (n-1);
Print n;
}

Derive the index formula for accessing elements in a 2-D array stored in row-major order

Array = Arr
No. of rows = R
No. of columns = C
Address of Arr = &Arr

Know that — total elements = R × C

Arr[i][j] = &Arr + (i × C + j) × sizeof(data)
↓ no. of row ↓ no. of column

Page No.

Saathi

Date

Ques6 Discuss how sparse matrices are represented and explain any one representation method.

Ans) A sparse matrix is a matrix in which most of the elements are zero.

→ Triplet representation :-

(Rownumber, Column no:, value)

$$\begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 9 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

0	2	3
2	0	5
2	3	9
3	1	2

Ques7 Explain the process of reversing a singly linked list with an ex?

Ans Reversing a singly linked list means changing the next pointers so that the head node becomes the last node.

```
listnode* reverselist (ListNode* head ){
    ListNode * prev = Null;
    List Node * curr = head;
    List Node * next = Null;
      while (curr != Null ){
        next = curr → next
        curr → next = prev;
        prev = curr;
        curr = next;
    } return prev;
```

Page No.

```
void insertion sort (int A[], int n) {
    for (int i = 1; i<n; i++) {
        int key = A[i];
        int j = i-1
        while (j>=0 && A[j] > key){
            A[j+1] = A[j];
            j--;
        }
        A[j+1] = key;
    }
}
```

Working → [5 2 4 6 1]

1 → [2, 5, 4, 6, 1]
2 → [2, 4, 5, 6, 1]
3 → [2, 4, 5, 6, 1]
4 → [1, 2, 4, 5, 6] = Answer

Time Complexity →

TC => $O(n^2)$.

Saathi

Date / /

Ques 4) Explain the difference between linear and binary search.

Ans) Linear search : One by one checks all elements in array.

Binary search : works on sorted array by repeatedly dividing the search interval in half.

Linear Search	Binary Search
→ starts from first element and search one-by-one	compare key with the middle element → if equal → found → if smaller → left ✓ → if greater → right ✓
→ TC → O(n)	TC → O(log n)
→ works on sorted + unsorted	works on sorted.

Ques 5) Write algorithm and explain insertion sort with an example.

Ans) Insertion sort is a simple sorting technique that places each element from the unsorted part i its corrected position in the sorted part of the array - similar to sorting playing cards in min

Page No.

Date / /

Saathi

→ Merge Sort is faster and more efficient for larger data sets.

Ques 9/10:

Ans) A DLL has nodes connected in both direction for polynomial representation :-

→ Coefficient
→ power
→ Pointers : next & previous.

• Node Structure.

```
Struct Node {
    int coeff;
    int pow;
    Node* next;
    Node* prev;
    Node(int c, int p) {
        coeff = c;
        pow = p
        next = null;
        Prev = null;
    }
}
```

Date / /

Ques 9

Ans

```
Void mergesort (int arr[], int left, int right, int temp[])
  if (left < right) {
    int mid = (left + end-st)/2;

    mergesort (arr, left, mid, temp);
    mergesort (arr, midtl, right, temp);
    merge (arr, left, mid, right, temp);
  }
}

Void merge (int arr[], int left, int right, int mid, int temp)
  int i = left;
  int j = mid + 1;
  int K = left;

  while (i <= mid && j <= right) {
    if (arr[i] <= arr[j]) {
      temp[K++] = arr[i++];
    else
      temp[K++] = arr[j++];
  }

while (i <= mid) {
  temp[K++] = arr[i++]; }
while (j <= right {
  temp[K++] = arr[j++];

for (inp p = left; p <= right; p++) {
  arr[p] = temp[p]; }.
}
```

Page No.

Saathi

Date

Ques a) Write a recursive algorithm for the Tower of Hanoi problem and explain it.
b) Trade offs between recursion and iterations.

Ans a

```
Void TOH ( n , source , target, auxilary) {
    if n==1 then
        print "move disk 1 from" source "to" target
    else
        TOH (n-1 , source , auxilary , target)
        print "move disk", n, "from", source, "to", target
        TOH (n-1, auxilary , target, source);
```

number of moves $2^n - 1 = 7$

Ans b

	Recursion	Iterations
1)	function calls itself until a base condition is met	Repeats a set of statem using loops.
2)	uses a base condition	uses a loop conditi
3)	more memory due to function call stack.	No memory / les
4)	slightly slower	faster.

Set 2

Q1

Ans Time space tradeoff is a concept in algorithm design where we can reduce execution time by using more memory or save memory by allowing the algorithm to take more time. It represents the balance between speed and storage.

Example → • using hash table → memory ↑ Time ↓
using linear search → memory ↓ Time ↑

Q2

Ans A multidimensional array is an array that contains more than one index or subscript. It is used to represent data in a table like structure, such as matrices, grids or 3D data.

Applications :-

Matrices and mathematical
image processing.
Game development.
Data Tables.

```
int marks[3][3] = {{85, 90, 78},
                   {80, 88, 92},
                   {75, 84, 89}};
```

Page No.

Q3
Ans Derivation of Index formula for 1D array.

- BA = Base Address
- W = size of each element
- i = Index of the desired element.

Then;

arr[i] = BA + (i x w);

Ex arr[2] = BA + 2 x w
= BA + 2w;

Q4
Ans Linear Search is a simple searching technique wh each element in the array is checked one by one until the desired element (Key) is found the list ends.

- works on both sorted and unsorted arrays

```
int linearSearch (int arr[], int n, int key){
    for (int i=0; i<n; i++){
        if (arr[i] == key){
            return i;
        }
    return -1;
}
```

Saathi

Advantages :

i) Simple and easy to implement.
ii) Works for both sorted and unsorted data
iii) Suitable for small datasets.

Limitations :

i) Inefficient for large arrays.
ii) TC = O(n)
iii) SC = O(1)

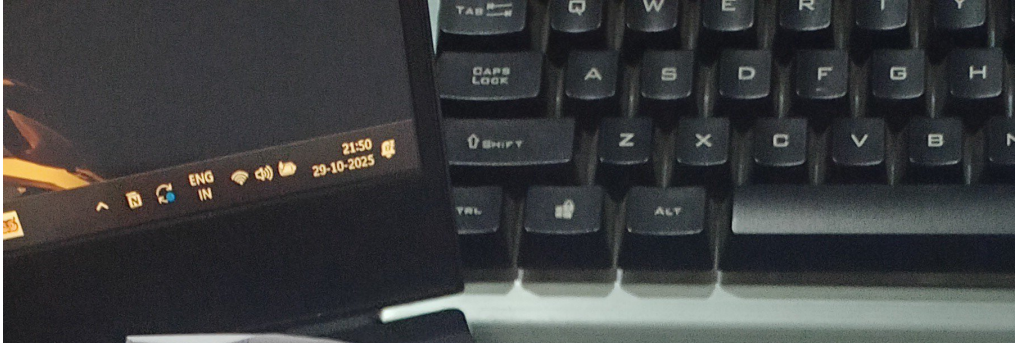
Q5

Ans) QuickSort is a Divide and Conquer sorting algorithm.

→ It selects a pivot element, then partioning element such as
- Elements < pivot are at left
- Elements > pivot are at right.

```
int partition(vector(int)& arr, int low, int high){
    int pivot = arr[low]
    int i = low;
    int j = high;
    while (i<=high && arr[i] <= pivot)
        i++;
    while (j>= low && arr[j] >pivot)
        j--;
    if(i<j) Swap (arr[i], arr[j]);
```

Page No.



Date ___/___/____

Saathi

Ques?

Ans A Doubly Linked list (DLL) is a linear data structure where each node contains three part.

1) Data.
2) pointer = prev node
3) pointer = next node.

Structure Node

```
Struct Node {
    int data;
    Node* prev;
    Node* next;
};
```

Null ← [Prev | 10 | next] ⇄ [Prev | 20 | Next] → N

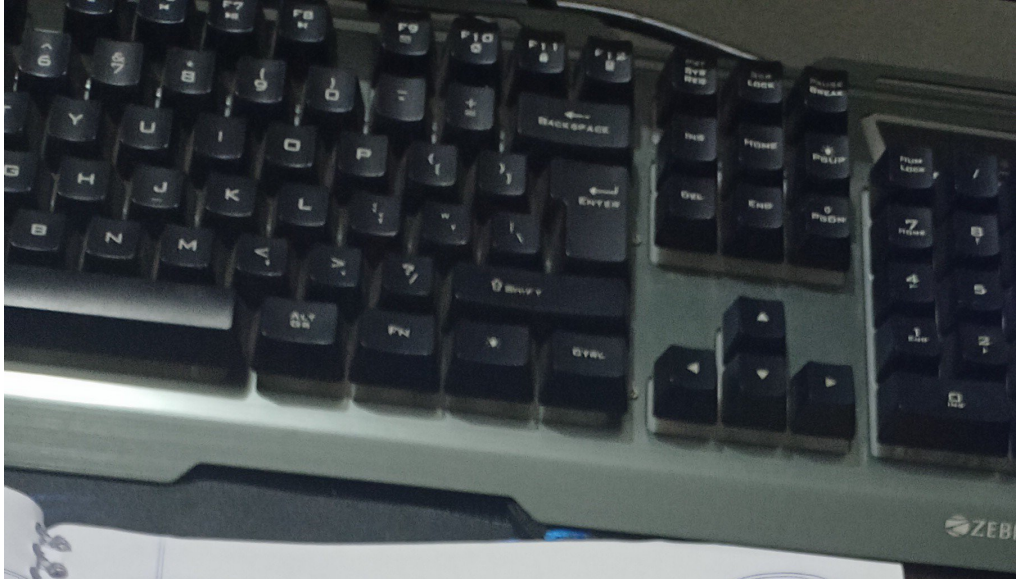
↳ Doubly Linked list.

Insertion in DLL.

i) at begining → insert (Node **Head, int val

```
Node *new = new node;
new → data = val;
new → prev = null;
new → next = * head;
if (* head != null)
    * head → prev = new
```

Page No.



Saathi

Date ___ / ___ / ___

b) Indirect Recursion.

```
Void A() {
    cout << A;
    B();
}
Void B() {
    cout << B;
    A();
}
```

c) Tail Recursion :

```
Void A (int n){
    if (n == 1) return;
    print n;
    A(n-1);
}
```

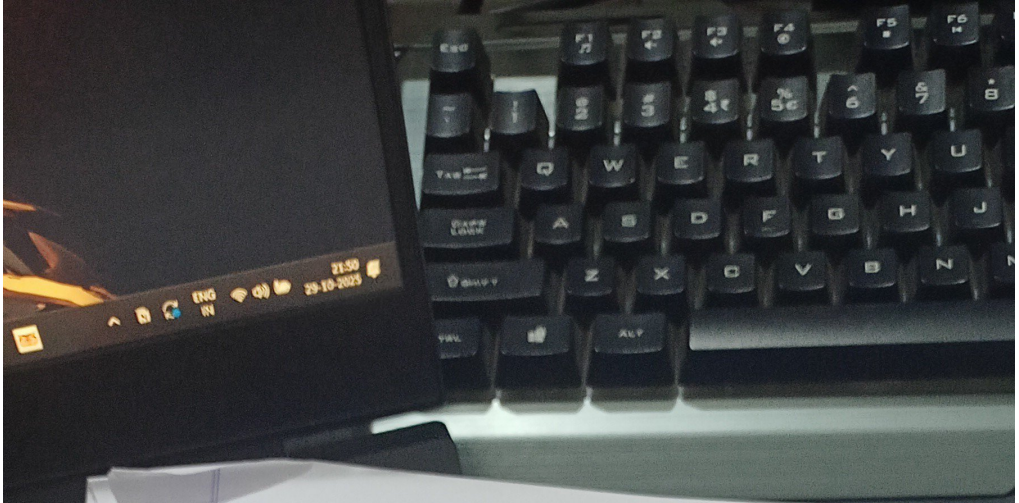
↳ call at last

d) Head Recursion

```
Void A (int n) {
    if (n == 1) return;
    A(n-1);
    print n;
}
```

↳ call at first

Page No.



Date ___/___/_____ Saathi

```
    swap(a[low], a[j]);
    return j;
}

void qs(vector<int> &arr, int low, int high){
    if(low<high){
        int pIdx = partition(arr, low, high);
        qs(arr, low, pIdx-1);
        qs(arr, pIdx+1, high);
    }
};
```

TC = O(n log n);

Ques 6

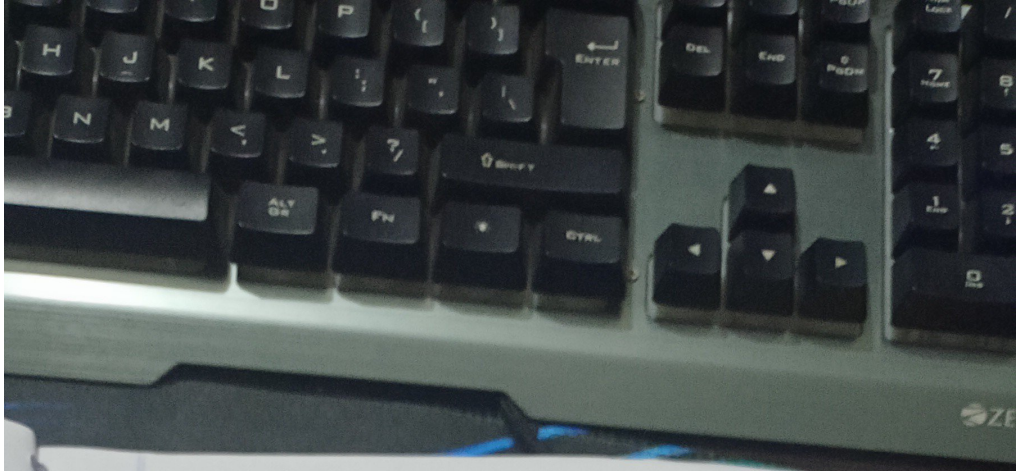
Ans Recursions is program in where a function calls itself directly or indirectly.

Types of recursion :

→ Direct recursion

```
void fun(){
    cout << "Hello";
    fun();
}
```

Page No.



Saathi

Date / /

```
    *head = new;
}
```

Ex insert(5)

Null ← [10] ⇄ [20] ⇄ [30] → null

↓

Null ← [5] ⇄ [10] ⇄ [20] ⇄ [30] → Null.

⇒ at the end ;

~~Create~~ insert (Node** head, int val) {
Node* new = new Node;
~~while~~
Node* temp = *head;
while (temp→next != null)
temp = temp→next;

temp→next = newnode;
newnode→prev = temp;
newnode→next = null;
}.

→ Deletion → It is same as insertion we just need to know how to play with the pointers.

( )

Page No.

Ans: Fibonacci with iterations

```
Void fabinacci (int n) {
    int a=0, b=1, c;
    cout << a << b;
    for (int i= 2; i<n; i++){
        c= a+b;
        cout<<c;
        a = b
        b = c;
    }
}
```

b) Removal of Recursion :-

- Recursion removal means converting a recursion function into iterative
- It reduces function call and stack memory.

Saathi

Ans 9

Ans) A sparse matrix is one in which most elements are zero and only a few are non zero.

Eg: $A = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 0 \\ 5 & 0 & 0 \end{bmatrix}$

→ Linked List representation :

- Row index
- Column index
- Value
- pointer to next non zero element

→ Structure :

```
struct Node {
    int row, col, value;
    Node* next;
};
```

Date / /

Saathi

Set - 3

Q1

Ans) Algorithm efficiency means how well an algorithm utilizes the time and memory resources to solve a problem.

Measurement :

1) Time complexity → The amount of time an algorithm takes to run
2) Space complexity → The amount of memory an algorithm takes during execution.

They are analyzed using asymptotic notation.

Q2

Ans)

	Time Complexity	Space complexity
i)	It measures the time of execution	It measures the space used.
ii)	depends on no. of operation	depends on variables, data structures etc.
	Linear search $TC = O(n)$	Recursive Fibonacci $O(n)$.

Q3

Ans A 3-D array can be visualized as an array of 2D matrices

```
int A[L][M][N];
```

L = number of planes
M = Number of rows
N = Number of columns.

BA = Base Address.
W = size of 1 element.

∴ A[i][j][k] = BA + W (k × (l × m) + j × l + i)
Column major order.

~~Ans.~~

Q4

Ans Binary search is an efficient searching algorithm used to find an element in a sorted array.

- It divides searching by half at each step.
- works only on sorted data.

Date ___/___/___

Saathi

```
Initialize low = 0 high = n-1;
while (low <= high) do
    mid = (low + high) / 2.
    if arr[mid] == key then
        return mid.
    else if arr[mid] < key then
        low = mid + 1
    else
        high = mid - 1.
end while;
return -1.
```

1- find first mid element.
2- if mid == key → found.
3- if key is smaller → search left half
4- if key is greater → search right half
5- Repeat until found or range is empty.

Ques

Ans) Bubble sort is a simple comparison techniq in which after 1 iteration and adjacent swapping the greatest element bubbles up at the end

```
Bubblesort (int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
                swap (arr[j], arr[j+1]
        }
    }
```

Date

Saathi

Ex = { 5, 2, 8, 4, 1 }

After 1st iteration

{ 1, 5, 4, 1, 8 }
↓
{ 2, 4, 1, 5, 8 }
↓
{ 2, 1, 4, 5, 8 }
↓
{ 1, 2, 4, 5, 8 }

Ques

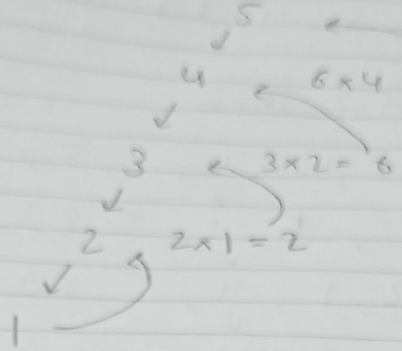
Ans) The factorial of a number "n" is the product of all positive integers less than or equal to n

$n!$

Ex = $5! = 5 \times 4 \times 3 \times 2 \times 1$

```
int factorial (int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial (n-1);
}
```

Page No.



Q.

Ans A Circular Linked List (CLL) is a version of Linked List where the last node points back to the first node, forming a loop.

* Representation:

[10|*] → [20|*] → [30|*]

```
Struct Node {
    int data;
    Node * next;
}
```

} structure.