DSA:

4M:

① Explain array and its operations?
② Polynomial arithmetic?
③ Towers of Hanoi?
④ Explain dynamic data structure and its operations!
⑤ Why are we using cursor implementation in linked lists?
⑥ Explain about infix, prefix, postfix, Comin in (2M also).
⑦ Josph problem (Sparse Matrix).

12M:

① Explain Linked-lists - Singly, doubly, Circular, Linked lists & its operations.
② Explain Sparse matrix.
③ Explain Stack array and Stack Linked lists.
④ Explain queue array and queue linked lists.
⑤ Explain double ended queue and priority queue (applications of priority queue included).

ANSWERS.

4M

① Explain array and its operations:

*Array: → It is a collection of data items having similar data types stored in contiguous memory locations.

* Operations of array: (Basic operations):

→ Traverse → print all the array elements one by one.
→ Insertion → adds an element at the given index
→ Deletion → Deletes an element at the given index
→ Search → Searches an element using the given index of or by the value.

Update → Updates an element at given index.

*Imp terms to understand concept of array:

→ Element: each item stored in an array is called an element.
→ Index: Each location of an element in an array has a numerical index, used to identify the element.

② Polynomial Arithmetic:

→ The manipulation of symbolic polynomials, has a classic example of list processing. In general, we went to represent the polynomial.

$A(x) = a_{m-1} x^{e_{m-1}} + \cdots + a_3 x^{e_0}$

→ We will represent each term as a node containing coefficient and exponent fields as well as a pointer to the next term. A node structure for a polynomial is shown below.

coef	expon	link

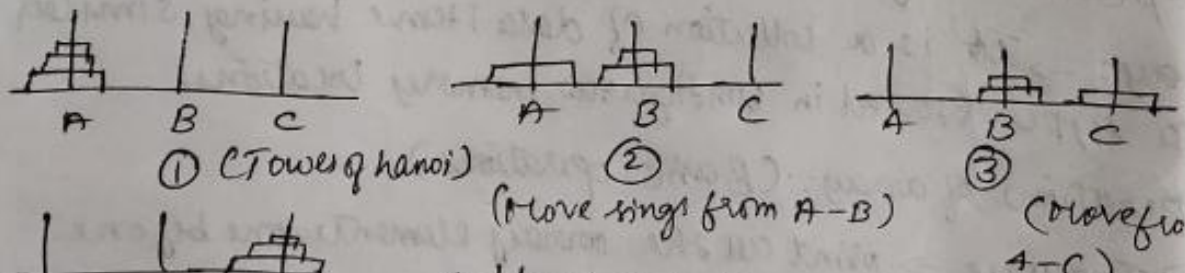
→ Assuming that the coefficients are integer, the node structure will be declared as follows:-

```
struct link
{
    int coeff;
    int row;
    struct link * next;
};
```

③ Towers of Hanoi:

→ The towers of hanoi is one of the main application of recursion.

→ If you can solve n-1 cases, then you can easily solve the nth case.



① (Towers of hanoi)

② (Move rings from A-B)

③ (Move for A-C)

④ (Move ring from B-C)

* Here: A → Source pole
B → Spare pole
C → destination pole.

* Base case if n=1 → move the ring from A-C using B as spare.

* Recursive case:

→ Move n-1 rings from A-B using C as spare.

→ Move the one ring from A-C using B as spare.

→ Move n-1 rings from B-C using A as spare.

④ Dynamic data structure and its operations :-

→ Dynamic data structure is that kind of data structure that changes its size during run time.

→ The value stored in data structure can be changed easily either its static or dynamic data structure.

→ But dynamic data are designed in such a way that both data and size of data structure can be easily changed at the run time.

* Major examples of dynamic data structures are:

→ Single linked list
→ Double linked list
→ vector
→ Stack
→ Queue
→ Tree.

* Operation of dynamic D-S.

→ Traversal → Sorting.
→ Insertion
→ deletion
→ Searching
→ updating

⑤ Why are we using cursor-based implementation in linked lists?

* Cursor-based implementation:

→ If linked lists are required and pointers are not available, then an alternate implementation must be used.

→ The alternative method we will describe is known as Cursor implementation.

→ The two important items present in a pointer implementation of linked lists are:

→ The data is stored in collection of structures, Each structure contains a data and a pointer to next structure.

→ A new structure can be obtained from the system's global memory by a call to malloc and released by a call to -

→ Q2
→ Q4
→ Q5
→ Q7 free

⑥ Explain about infix, prefix, postfix:-

→ Infix notation
→ Prefix notation
→ Post fix notation

} The way to write arithmetic expression is known as notation the three types are as follows.

* Infix notation:

→ We write expression in infix notation where operators are used inbetween operands. e.g: a - b + c;

→ It is easy for humans to read, write & speak in infix notation but the same does not go well with computing devices.

→ An algorithm to process infix notation could be difficult and costly in terms of time and space consumption.

* Prefix Notation:

→ In this notation, operators are prefixed to operands, that is i.e. operators are written ahead of operands.
ex: +ab.

→ This is equivalent to its infix operation a+b.

→ Prefix notation is also known as polish notation.

* Post fix Notation:

→ This notation style is known as reversed prefix or polish notation.

→ In this notation style, the operator is post fixed to the operands i.e. the operator is written after the operands.

→ ex: ab+ is equivalent to a+b.

⑦ Joseph problem: (Sparse matrix).

→ A matrix can be defined as a two-dimensional array having 'm' columns, and 'n' rows representing m * n matrix. Sparce matrices has majority of their elements equal to zero. In other words, the sparse matrix can be defined as the matrix that has a greater number of zero elements than the non zero elements.

→ there are two advantages of sparse matrix:

→ Storage.

→ Computing time.

* Storage:
→ As we know a sparse matrix that contains lesser non-zero elements than zero, so less memory can be used to store elements.
→ It evaluates only the non-zero elements.

* Computing time:
→ In the case of searching a sparse matrix, we need to transverse only the non-zero elements rather than traversing all the sparse matrix elements.

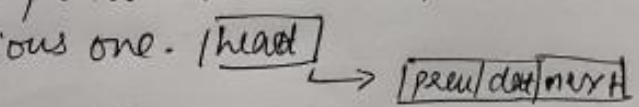
* The Sparse matrix can be represented in two following ways:
→ Array representation:
→ Linked list representation:

	0	1	2	3
0	0	0	1	2
1	3	0	0	0
2	0	4	5	0
3	0	6	0	0

→ Sparse matrix.

(8) Singly linked list:
→ It is a type of linked list that is unidirectional, i.e. it can be traversed in only on direction from head to the last node.
→ Each element in a linked list is called a node. A single node contains data and a pointer to the next node which helps in maintaining the structure of list.
→ Operations of singly linked list: → Node generation, Insertion deletion & traversing.

* Doubly linked list:
→ It is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.
→ Therefore in a doubly linked list, node consists of three parts : node data, pointer to the next node in sequence, pointer to the previous one.



* Circular singly linked list: → In this, the last node of the first contains a pointer to the first node of the ~~list~~ list.
→ We can have circular singly as well as circular doubly LL.
→ We traverse a circular singly linked list until we reach the same node where we started. It has no start & no end.