DATA STRUCTURES

- 7 MARK ANSWERS -

1. Define linked list. Explain the operations of linked list and write its advantages and disadvantages.

Definition:

Linked list is the non-sequential linear data structure that has a collection of nodes. Each nodes have categorised into data fields and linked fields.

* Data fields stores the data and information that are required to store in a node.
* linked field represents the address of the node and are used toas a pointer to point to the next node.

Operations:

Premitively, there are two operations. One is insertion and another one is deletion.

→ Insertion, to insert a node at specified position.

→ Deletion, to remove the node from list at specified position.

Mechanisms for linked list:

1. A mechanisms require to frame chunk of memory ~~that~~ into nodes ~~that~~ with required no. of data and links.
2. Determine which nodes are free and which nodes are needed to be allocated.
3. Obtain a required no. of nodes from the free storage area. [GETNODE(x)]
4. Remove/Return or dispose the data item to the free pool area. [RETURN(x)].

Advantages of linked list:

1. Linked list are dynamic data structure. That is they ~~ca~~ can grow or strink during the execution of program.

2. Efficient memory allocation: The nodes for the linked list cannot be preallocated into memory. They can allocate whenever required and de-allocated whenever not needed.

3. Insertion and deletion of nodes is done more effectively. ~~They~~ The ~~can~~ node can ~~dete insert~~ be ~~nodes~~ inserted ~~ff~~ more flexibly to the specified position and can deleted from the ~~me~~ linked list whenever required.

4. More complexive applications are done effectively and easier with the help of linked list.

(Disadvantages) Limitations of linked List:

1. Need More space: ~~We~~ They need more space ~~to the~~ require for storing a data.

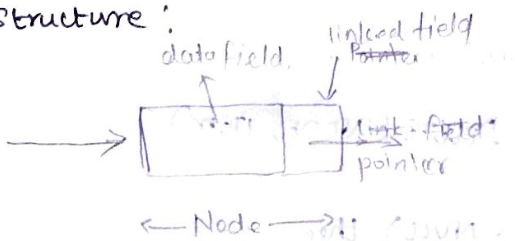
2. Access of arbitrary data item is little bit cumbersome and time consuming because ~~they~~ they purely supports the sequential traversal.

2. What is the

2. What is the Singly linked List ? Write down its operations.

Linked list (singly linked list) is the linear collection of ~~data~~ specially designed data elements, called nodes and the ~~pointer~~ link that points to the another by the means of pointer.

Basic Structure :



Data field: to store imformation into a node.

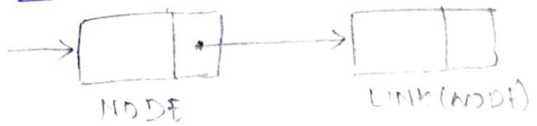
Link field: Represents the address of node and pointer to points to another node.

Premitive operations :

(i) Insertion algorithm:

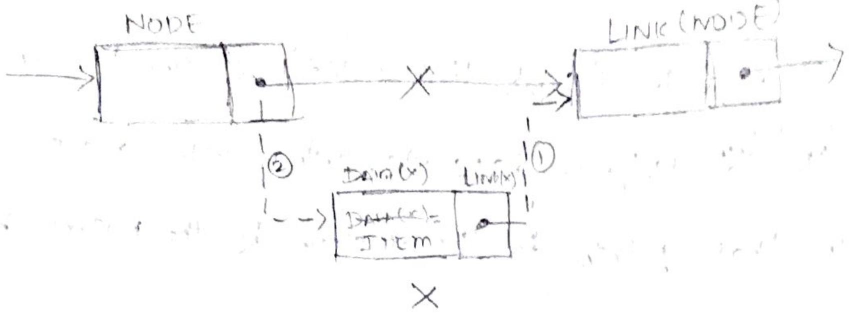
```
Procedure INSERT(START, NODE, ITEM)
{
call GETNODE(x);
DATA(x) = ITEM;
 if(START == NULL) then
     LINK(x) = NULL;
     START(x);
 else {
    LINK(x) = LINK(NODE);
    LINK(NODE) = x;
 }
 end INSERT.
```

Before Insertion:



After

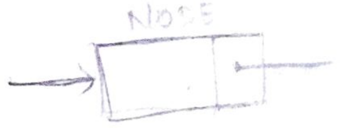
After insertion:



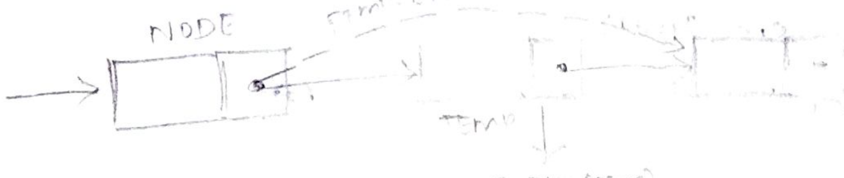
(ii) Deletion algorithm:

```
Procedure DELETE (NODE, ITEM)
{
  if (NODE == NULL) then
      Call ABANDON_DELETE;
  else {
      TEMP = LINK (NODE);
      LINK (NODE) = X;
      RETURN (TEMP);
  }
}
end DELETE:
```

Before deletion:



After deletion:



3. What is Circularly linked list? Give the operations for it.

Circularly linked list is a linked list data structu
where the ~~data~~ movement of data elements (node) design is circular instead of linear. Here the last pointer points to links to the first node.

Operations of Circular list

(i) **Insertion algorithm**:

(a) Left most insertion:

```
Procedure INSERT_CLL(P,X,A)
Call GETNODE(X);
DATA(X) = A;
LINK(X) = LINK(P);
LINK(P) = X;
End INSERT_CLL.
```

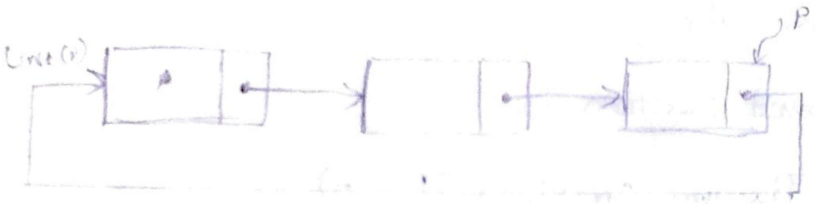
Before insertion:

After Insertion:

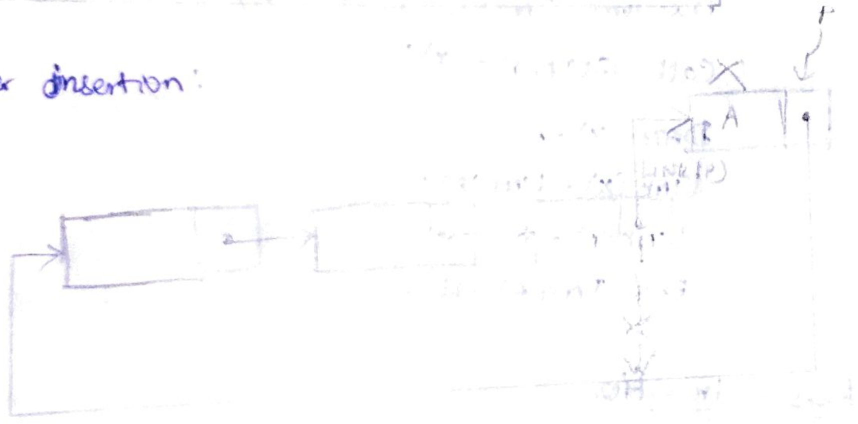
(b) Right most insertion:

```
Procedure INSERT_CLL (P,X,A)
call GETNODE(X);
DATA (X) = A;
LINK (P) = X ~~LINK (x~~);
P = LINK (x) ~~x~~;
end INSERT_CLL
```

Before insertion:



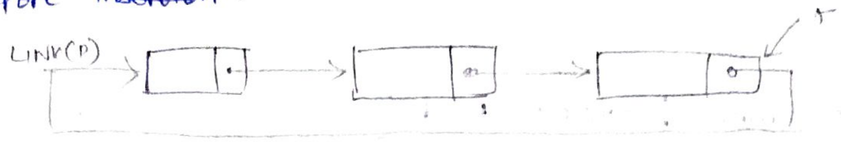
After insertion:



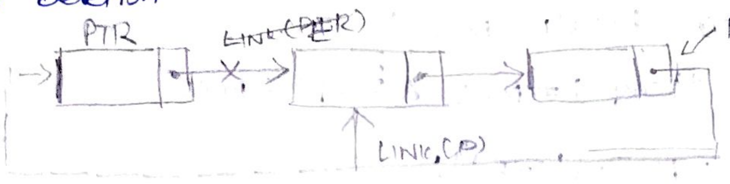
(ii) Deletion Algorithm:

```
Procedure DELETE_CLL (P)
PTR = LINK (P);
Y = DATA (PTR);
LINK (P) = LINK (PTR);
call RETURN (PTR);
end DELETE_CLL.
```

Before ~~Insertion~~ deletion :



After Deletion

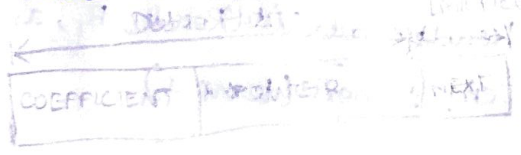


4. Write down the applications of linked list ?

Addition of Polynomials

Let P_1 & P_2 be the given polynomial, to add P_1 and P_2, these are first arranged on the descending order of the exponents and corresponding variable x.

The node structure required for this task,



Example :

$P_1 \rightarrow 7x^6 + 8x^5 - 9x^3 + 10x^2 + 14.$

$P_2 \rightarrow 2x^6 + x^3 + 5x + 4.$

Then $P_1 + P_2 = 9x^6 + 8x^5 - 8x^3 + 10x^2 + 5x + 18.$

To perform this symbolic manipulation of polynomials we use singly linked list whose structure is given above.

Steps :

① Presuming that they are arranged in descending order of exponents.

② To acheive $P1+P_2$, we add the coefficient field of node of like powers (~~power~~ exponent 1 = exponent 2) of x in $P_1 + P_2$ list.

③ The resultant is done by creating new node and storing results.

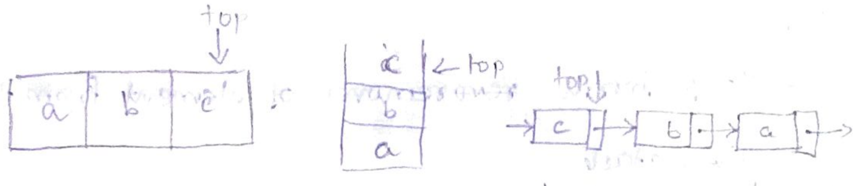
Algorithm:

```
Procedure POLYADD($P_1$, $P_2$)
(1) initialise ~~B~~ temp1 = $P_1$; temp2 = $P_2$;
(2) initiate Result = NULL;
(3) while (temp1 $\neq$ NULL & temp2 $\neq$ NULL)
        $a_1$ = power (temp 1);
        ~~A~~ $a_2$ = power (temp 2);
        $B_1$ = coeff (temp1);
        $B_2$ = coeff (temp2);
        if ($a_1 = a_2$) then
            if ($b_1 + b_2 \neq 0$) then
                Result = addlist (Result, $B_1 + B_2$, $a_1$)
                temp1 = next (temp1)
                temp 2 = next (temp 2)
        else if ($a_1 > a_2$) then
                Result = add list (Result, $B_1$, $a_1$)
                temp 1 = next (temp 1)
        else
            Result = addlast (Result, $B_2$, $a_2$)
            temp 2 = next (temp 2)
        end while
(4) if (temp 1 $\neq$ null) then
        while (temp 1 $\neq$ null)
            Result = addlast (Result, $B_1$, $a_1$);
            temp 1 = next (temp 1)
        end while
    else if (temp 2 $\neq$ null) then
        while (temp $\neq$ null) { Result = addliast (Resut, $B_2$, $a_2$);
                                   temp 2 = next (temp 2); }
        end while
    Return (Result)
End POLYADD.
```

) Define linked stack. Explain with its operation.

* Stack usually inserts or deletes an elements from the 'top'. That is the last that comes in will be the first to go out (LIFO).

* Linked stack is the linear list of elements commonly ~~used~~ implemented ~~for~~ use as a singly linked list, on which the ~~fi~~ pointer points to the top of the stack or first node.



Stack Representation

Linked list representation

<u>Operations</u>:

The operations are ~~the~~ same as the ordinary linear stack.

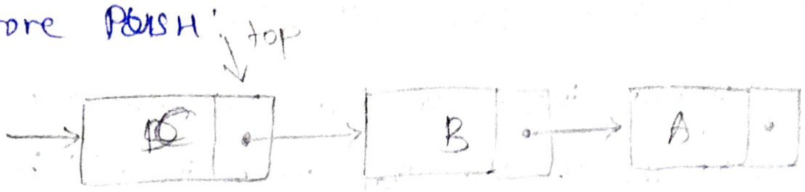
(i) Push:

→ It performs the insertion of element to the top of the stack.

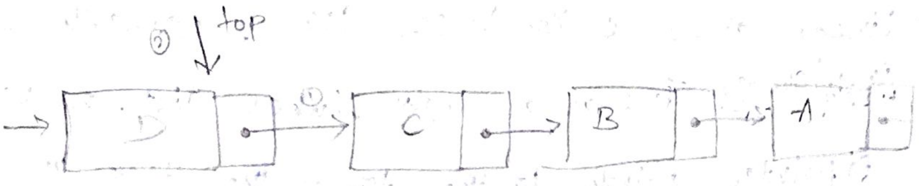
algorithm: ~~Pro~~

```
        Procedure PUSH_LS (TOP, ITEM, x)
    Call GETNODE(x);
        DATA(x) = ITEM;
        LINK(x) = TOP;
        TOP = x;
        End PUSH_LS.
```

Before PUSH :



After PUSH :- LS(TOP, D, X)



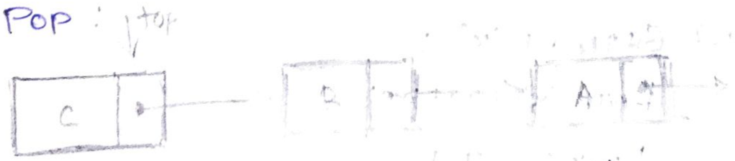
(ii) Pop

→ It performs ~~[illegible]~~ removel of element from the top of the given stack.

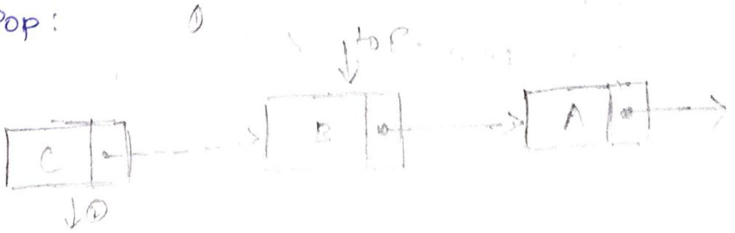
Algorithm :

```
Procedure Pop(TOP, ITEM)
if TOP == NULL then Abandon_stack ;
else {
    TEMP = LINK (TOP) ;
    ITEM = DATA (TOP) ;
    ~~LINK (TEMP) =~~
    TOP = LINK (TOP) ;
    RETURN (TEMP) ;
  }
End Pop.
```

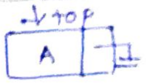
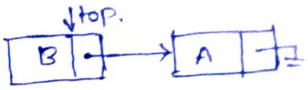
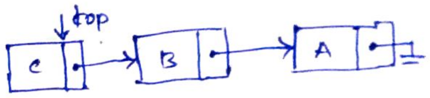
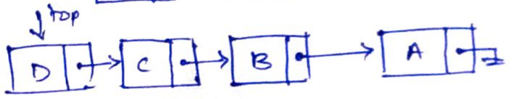
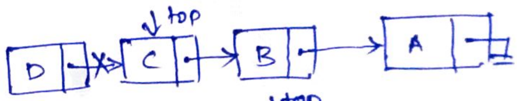
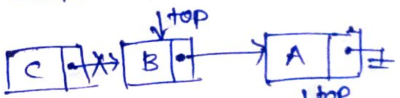
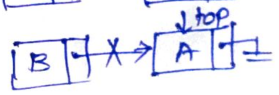
Before Pop :



After Pop :



Example :

	Invokations	Linked Stack Representation	~~Bx~~ Result
1.	~~Push~~ (Existed.		—
2.	PUSH (TOP, B, X)		insertion completion
3.	PUSH (TOP, C, X)		insertion completion
4.	PUSH (TOP, D, X)		insertion completion
5.	POP (TOP, D)		deletion completion
6.	POP (TOP, C)		deletion completion
7.	POP (TOP, B)		deletion completion