PSZ- DUET Admission coaching (20)

Data Structure & Algorithm

* **Data Structure:** কোনো ডাটা ব্যবহারে ডাটার লজিক্যাল ও গাণিতিক মডেলকে Data structure বলে।

Data Structure প্রধানত 2 প্রকার:

(i) Linear Data structure

(ii) Non-Linear Data structure

* **Linear Data Structure:** যে Data structure এর উপাদান সমূহ পর্যায়ক্রমিকভাবে সাজানো থাকে তাকে Linear Data Structure বলে। যেমন: Array, stack, Linked List, Queue, pointer

* **Non-Linear Data Structure:** যে Data structure এর উপাদান সমূহকে কোন নির্দিষ্ট ক্রমানুসারে সাজানো হয়না তাকে Non-Linear Data structure বলে। যেমন: Tree and Graph.

** Linear Data Structure এর Operation:

(i) Traversing: List এর প্রতিটি উপাদানে প্রবেশ প্রয়োগ করা।

(ii) Searching: List হতে কোন নির্দিষ্ট উপাদান খুঁজে বের করা

(iii) Insertion: List এ নতুন উপাদান সংযোজন করা

(iv) Deleting: List হতে কোন উপাদান বাদ দেওয়া

(v) Sorting: List এর উপাদান সমূহকে ক্রমানুসারে সাজানো

(vi) Merging: দুটি List কে একত্রিত করে একটি List তৈরি করা।

* বিভিন্ন Data structure এর Operation:

Operation on stack

i) Push
ii) Pop
iii) Make empty
iv) Isfull

Operation on Queue

i) Enque
ii) Deque
iii) Make Empty
iv) Is Empty

PSZ- 01870 656 119 - 01935 25 1961

Operations on Linked list.

i) Insert
ii) Delete
iii) Update
iv) search

Traverse methods on Tree

i) preorder
ii) Inorder
iii) Post Order

Graph Visiting Methods

i) Depth first Search (DFS)
ii) Breadth first Search (BFS)

** File → STUDENT

Field

Record →

S-ID	Name	F.Name	M.Name	Addr	Course	Year
01	Hasan	Noman	ABC	A	EEE	2006
02	Toma	Tahen	BCD	B	CSE	2008
03	Momi	Amin	EFG	C	CSE	2010
04	Tonu	Yousuf	Ma	D	ME	2012
05	Ma	Tofael	Mina	E	CI	2014

Cardinality = 4
Degree = 7.

*Field: একাধিক সংযুক্ত Data কে Data Item বলে। একাধিক Data item. Data সমূহকে Data Field বলে।

* Record
* File
* Degree
* Cardinality

DUET Admission coaching PSZ only for CSE Dept.

+ Data : Data বলতে কোনো নির্দিষ্ট বা set of value কে বুঝায়।
+ Information: Data কে process করলে Information পাওয়া যায়। Information হলো অর্থবহ Data.

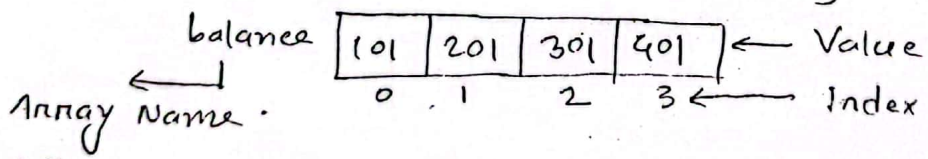
** Array: Array শব্দের আভিধানিক অর্থ শ্রেনীবদ্ধভাবে সাজানো। একই type এর অনেকগুলো Data র সমষ্টি যা Memory তে পরপর যতগুলো byte allocate করে তাকে Array বলে।

→Declaring Array: Data_Type Array_Name [size];
→Initial Array: int balance [5] = {2,4,6,8,10};
→Accessing Array Elements: int balance = sal = balance[

** Types of Array: Array সাধারনত 2 প্রকার:

① Linear Array/ One dimentional Array: একটি subscript ব্যবহৃত হয়।

int balance[4] = {101, 201, 301, 401}



OR

balance[0] = 101;
balance[1] = 201;
⋮
balance[3] = 401;

② Non-Linear/ Multidimentional Array: যে Array দুই বা ততোধিক subscript ব্যবহৃত হয়।
2টি subscript ব্যবহৃত হলে Two-Dimentional Array বলে।
syntax: Data_Type Array_Name [Row size] [Column s]
int Num [3] [4];

n- Dimentional Array :

Data-Type Array-Name [s1] [s2] [s3] . . . . [sn]

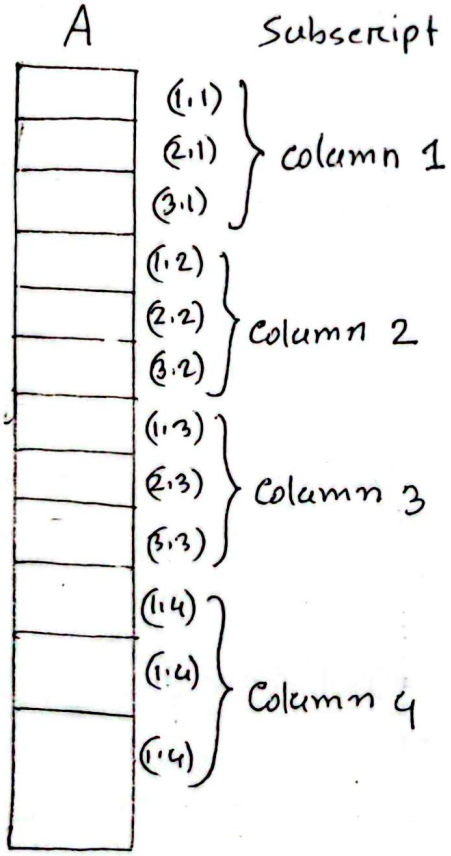
int num [3] [5] [2];

** Two-Dimentional Array

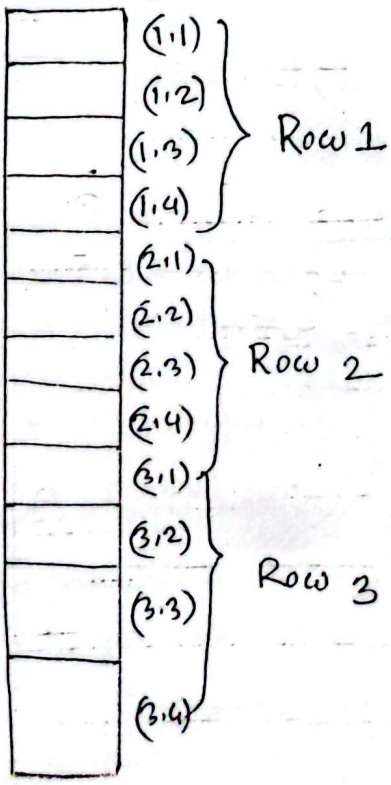
Rows \ Columns	1	2	3	4
1	A[1,1]	A[1,2]	A[1,3]	A[1,4]
2	A[2,1]	A[2,2]	A[2,3]	A[2,4]
3	A[3,1]	A[3,2]	A[3,3]	A[3,4]

** Two-dimentional 3X4 array A

Representation of Two- Dimentional Array in Memory



Column-major order



One-dimentional Array, Length = UB − LB + 1

* UB is the largest index, called the upper bound
* LB is the smallest index, called the lower bound

*** Automobile company 1932 সাল থেকে 1984 সাল পর্যন্ত প্রত্যেক বছরে কত Automobile সংখ্যার Record Auto নামক Array তে সংরক্ষন করে। উক্ত Array টির উপাদান সংখ্যা কত

We know,

Length = UB − LB + 1
= 1984 − 1932 + 1
= 53

Here,
UB = 1984
LB = 1932

Loc(LA[K]) = Base(LA) + w (K − lower bound)

*** একটি Automobile company 1932 সাল থেকে 1984 সাল পর্যন্ত প্রত্যেক বছরের বিভিন্ন Automobile সংখ্যার Record Auto Array তে সংরক্ষন করে। যদি Base (Auto) = 200 এবং w = 4 words/memory cell হয়, তবে 1965 সনের Array element Address নির্নয় কর ?

We know,

Loc(Auto [1965]) = Base (Auto) + W (K − LB)
= 200 + 4 (1965 − 1932)
= 332 (Ans:)

LB = 1932
K = 1965
Base(Auto) = 200
w = 4
Loc(Auto [1965]):

*** XXX (−10:10), YYY(1935 : 1985), ZZZ (35) তিনটি Array
a. প্রত্যেক Array এর element সংখ্যা নির্নয় কর
b. Let Base (YYY) = 400 এবং w = 4 words/memory cell for YYY তাহলে YYY [1942], YYY [1988] এবং YYY [1999] এ Address নির্নয় কর।

Length (XXX) = UB − LB + 1 = 10 − (−10) + 1 = 21

Length (YYY) = 1985 − 1935 + 1 = 51

length (ZZZ) = 35 − 1 + 1 = 35

(b)

Loc (YYY[1942]) = Base (YYY) + W(k − LB)
= 400 + 4 (1942 − 1935)
= 428

Given,
Base (YYY) = 400
W = 4 word/M.C

Loc (YYY[1977]) = Base (YYY) + W (k − LB)
= 400 + 4 (1977 − 1935)
= 568

Loc (YYY (1999)) = Base (YYY) + W (k − LB)
= 400 + 4 (1999 − 1935)
= 656

*** The condition of LA is AAA(5:50) BBB(−5:10) and CCC(18)

a) Find out the number of element in each array?
b) Let. Base (AAA) = 300. W = 4 for AAA. Find the Address of AAA (15), AAA (35) and AAA [55]

a. Length (AAA) = UB − LB + 1 = 50 − 5 + 1 = 46
Length (BBB) = 10 + 5 + 1 = 16
Length (CCC) = 18 − 1 + 1 = 18

b. Loc [AAA (15)] = Base (AAA) + W (K − LB)
= 300 + 4 (15 − 5)
= 340

LOC (AAA[35]) = 300 + 4 (35-5) = 420

LOC(AAA[55]) = is not possible, cause 55 exceeds 50

Two-Dimentional Array : M X N
Row ← → Column

LOC (A[J.K])

Column-major order LOC (A[J,K]) = Base (A) + W [M(K-1) + (J-1)]
Row-major order LOC (A[J,K]) = Base (A) + W [N (J-1) + (K-1)]

*** Consider the 25 x 4 matrix array SCORE
suppose Base (SCORE) = 200 and there are W=4 words per memory cell. Furthermore, suppose the programming language store two-dimentional arrays using row-major order. Then the address of SCORE[12,3] the third test of the twelfth student.

LOC (SCORE[12,3]) = 200 + 4 [4(12-1) + (3-1)]
= 384

* যদি $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 0 & -4 \\ 3 & 2 & 6 \end{bmatrix}$ হয় তাহলে A এবং B ম্যাট্রিক্স-এর গুনফল বের করুন।

soln:

$$A \times B = \begin{bmatrix} (1\times2)+(3\times3) & (1\times0)+(3\times2) & (1\times-4)+(3\times6) \\ (2\times2)+(4\times3) & (2\times0)+(4\times2) & (2\times-4)+(4\times6) \end{bmatrix}$$

$$= \begin{bmatrix} 11 & 6 & 14 \\ 16 & 8 & 16 \end{bmatrix}$$ (Ans:)

** Array ব্যবহারের সুবিধা ও অসুবিধা লিখ।

সুবিধা:

(i) Variable declaration এর পরিশ্রম হ্রাস পায়
(ii) একসাথে অনেক ডাটা ইনপুট করা যায়
(iii) ডাটাগুলো সেকোরিতে পাশাপাশি অবস্থান করে
(iv) প্রোগ্রাম নির্বাহ দ্রুত হয়
(v) প্রোগ্রামের জটিলতা হ্রাস পায়
(vi) প্রোগ্রামের আকার ছোট হয়

অসুবিধা:

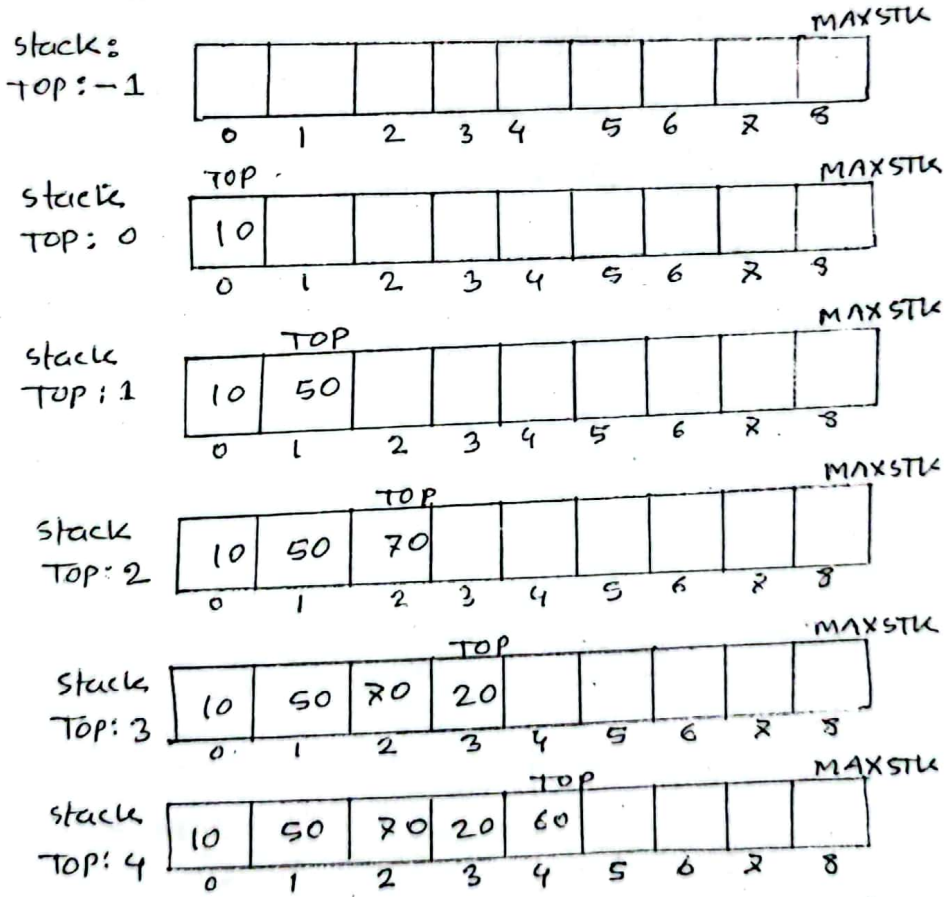
(i) প্রোগ্রাম নির্বাহের সময় অ্যারের মাপ পরিবর্তন করা যায়না
(ii) প্রকৃত ডাটা অপেক্ষা অ্যারের মাপ বড় হলে মেমোরি অপচয় হয়।

** MY PSZ বাক্যটি memory তে কত জায়গা দখল করবে 6byte. প্রতিটি character এর জন্য 1 byte এবং প্রতিটা space এর জন্য 1 byte ধরা লাগবে।

*** Stack এ DATA Insertion Operation: Stack এ Data Insert করার সময় দুটি বিষয় খুবই গুরুত্বপূর্ন। এর একটি হলো Top এবং অপরটি MAXSTK.
Top দ্বারা stack এ সর্বশেষ সংরক্ষিত Data র Address নির্দেশ করে এবং MAXSTK দ্বারা stack এর সর্বোচ্চ ধারণক্ষমতাকে অর্থাৎ stack এর Length size কে বুঝায়

** Stack এ নিচের ৫টি Data Insert করার step
10, 50, 70, 20, 60

Initial অবস্থা



*** DATA DELETION OPERATION: Stack এ Data item কে বিয়োজন process কে pop বলে। stack এর pop operation top থেকে শুরু হয়। stack এ data item অপসারনের জন্য অবশ্যই stack এ Data item থাকতে হবে। যদি top = -1 হয় তখন stack empty বুঝায় তখন deletion operation সম্ভব হয় না তার stack এর

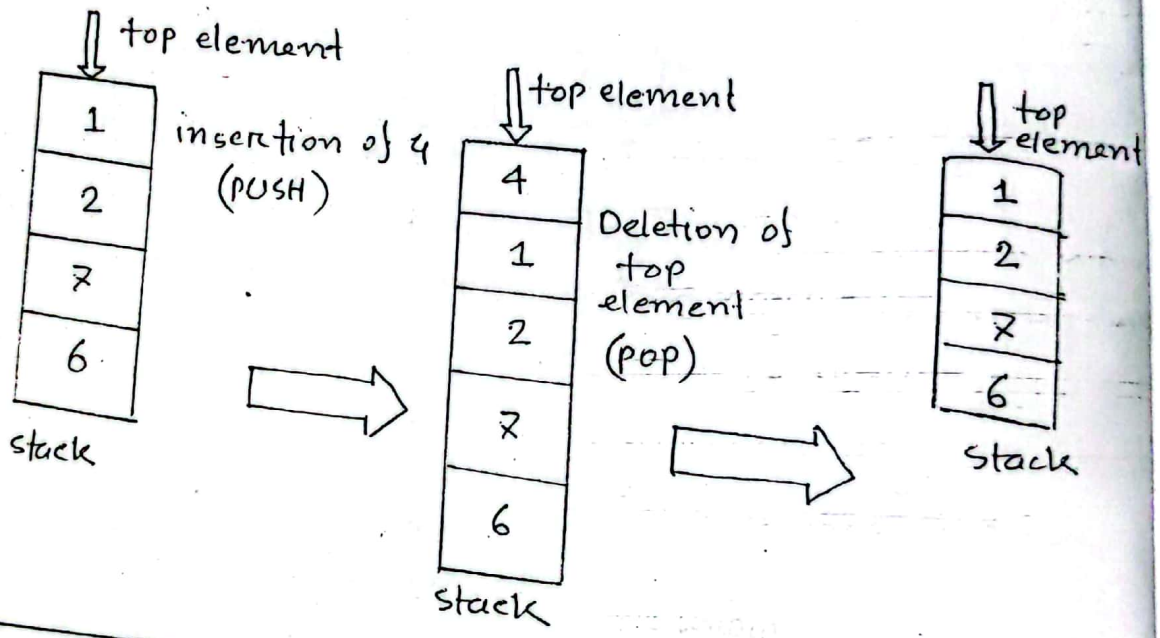
→

Stack

* Stack: Stack একটি Linear Data Structure. Linear ভাবে Memory তে Data store করে। এপদ্ধতিতে একের Data সাজানো হয় যাতে একটি স্তূপ তৈরি হয় এবং নতুন Data সংরক্ষণ বা সংরক্ষিত Data অপসারন শুধুমাত্র উপর হতে সম্ভব।

তাই একে LIFO অর্থাৎ Last-In-first-out পদ্ধতির Linear List বলা হয়। যেমন: plate এর stack.

A stack can be visualized as follows:



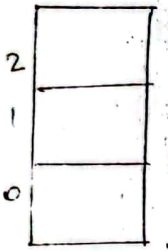
position of Top	status of stack
-1	Stack is Empty
0	Only One element in stack
N-1	stack is Full
N	Overflow state of stack

* Overflow
* Underflow
* PUSH()
* pop()

⑤ As top = 2, current size of stack is top+1, i.e 3
Now stack is full, as 3 is maximum size of stack

⑥ push(stack, 12, 3)
As stack is full, it will show
OVERFLOW CONDITION!

⑦ Deleting all elements from stack



Empty stack! ⟹ pop(stack, 3)
top = -1 pop(stack, 3)
pop(stack, 3)

⑧ pop(stack, 3)

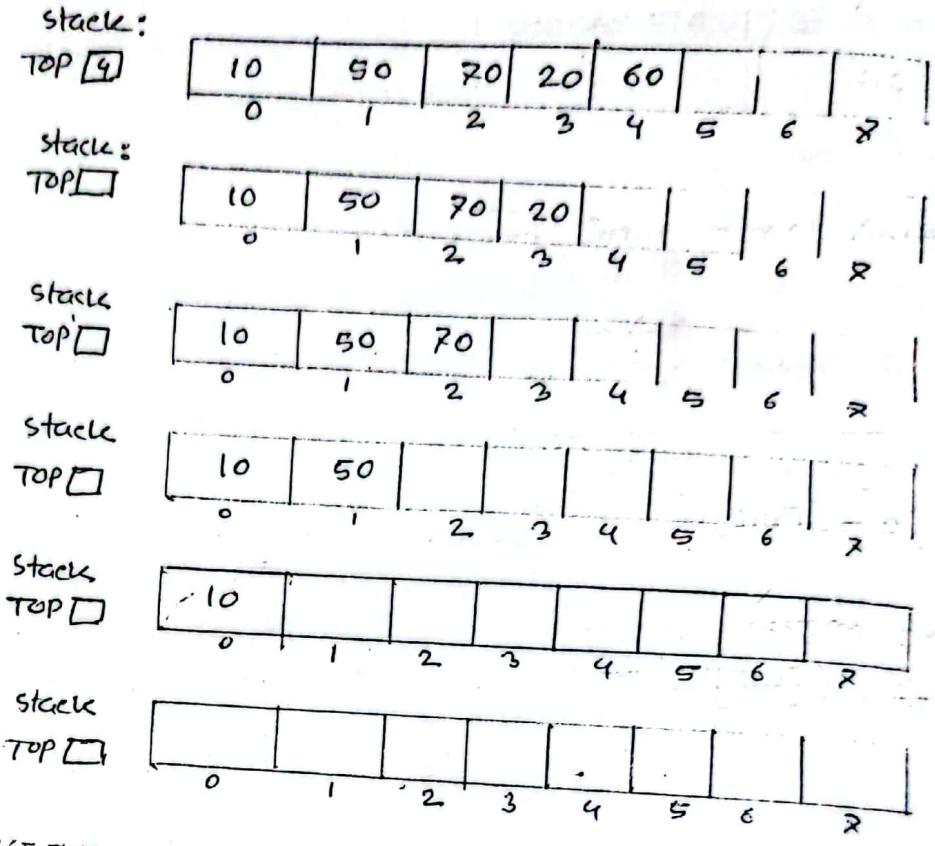
As, stack is empty. further deleting will cause
UNDERFLOW CONDITION!

*** Basic Operation on stack

- push() - pushing (storing) an element on the stack
- pop() - Removing (accessing) an element from the stack
- peek() - get the top data element of the stack, without removing it
- isFull() - check if stack is Full
- isEmpty() - check if stack is Empty

PSZ- 01870 656 119- 01935 25196]

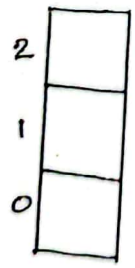
এই case কে Underflow বলে। Operation Process-



[বর্ননা ক্লাসে আলোচনা করলে নিচের থেকে দেখা যাবে]

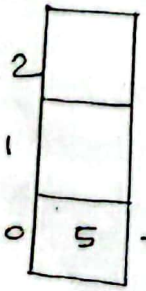
*** Another example of Insertion and Deletion in stack

①



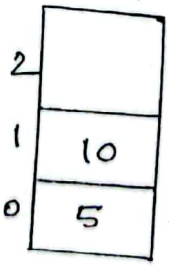
initial stack is empty
top = -1

② PUSH (stack, 5, 3)



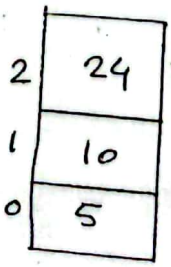
TOP = 0

③ PUSH(stack, 10, 3)



top = 1

④



TOP = 2

*** push operation/ Stack-এ Item Insert Algorithm

PUSH(stack, TOP, MAXSTK, ITEM)

STEP 1: (Is Stack Full ?)
if Top = MAXSTK, Then Print OVERFLOW and Return

STEP 2: TOP = TOP + 1

STEP 3: STACK(TOP) = ITEM

STEP 4: Return

*** pop operation/ stack থেকে Data Item বের করার

POP(STACK, TOP, ITEM)

STEP 1: (Is Stack empty ?)
if Top = 0 Print UNDERFLOW and Return

STEP 2: ITEM = STACK(TOP)

STEP 3: TOP = TOP - 1

STEP 4: Return.

* Stack এর Memory Representation দেখাও

Stack কে memory তে দুইভাবে represent করা যায়। যথা

① By using linear array

② By Using linear linked list

[বর্ণনা — প্রয়োগ — আলোচনা — করবে]

* Notation: The way to write arithmetic expression is known as a notation.

An arithmetic expression can be written in three different but equivalent notation, without changing the output.

These notation are —

- Infix Notation
- Prefix (polish) Notation
- Postfix (Reverse-polish) Notation.

[Details in class]

precedence and Associativity

Sn. No	operator	Precedence	Associativity
1	Exponentiation ^	Highest	Right Associative
2	Multiplication (*) Division (/)	second Highest	Left Associative
3	Addition (+) Substruction (-)	lowest	Left Associative

*** Infix Notation কে prefix ও Postfix এ পরিণত কর

① A/B*C - D+E/F/(G+H)

Prefix : - + - * / A B C / / E F + G H

Postfix : A B / C * D - E F / G H + / -

② ((A+B) * C - (D-E)) ↑ (F+G)

Prefix : ↑ - * + A B C - D E + F G

Postfix : A B + C * D E - - F G + ↑

③ (A+(B*C-(D/E↑F)*G)*H)

Postfix : A B C * D E F ↑ / G * - H * +

Postfix : H.W

*** Stack এর সাহায্যে মান নির্ণয় করঃ

① Infix : 5 + 3 ↑ 2 - 8 / 4 * 3 + 6

postfix : 5, 3, 2, ↑, 8, 4, /, 3, *, -, +, 6, +

Scanned	Stack
5	5
3	5, 3
2	5, 3, 2
↑	5, 9
8	5, 9, 8
4	5, 9, 8, 4
/	5, 9, 2
3	5, 9, 2, 3
*	5, 9, 6
-	5, 3
+	8
6	8, 6
+	14
)	

~~② postfix: 5, 6, 2, +, *~~

H.W → 5, 6, 2, +, *, 12, 4, /, - (37)

H.W → 12, 7, 3, -, /, 2, 1, 5, +, *, +

* Postfix Notation এর মান বের করঃ

P = 5, 6, 2, +, *, 12, 4, /, -

= 5, (6+2), *, 12, 4, /, -

= 5, 8, *, 12, 4, /, -

= 40, *, (12/4), -

= 40 - 3

= 37

* P = 12, 7, 3, -, /, 2, 1, 5, +, *, + Expression এর

① Infix Expression এ দেখাও

P = 12 / (7-3) + 2 * (1+5)

② P = 15 মান নির্ণয় করঃ

③ Stack এর সাহায্যে মান নির্ণয় করঃ

PSZ-1970 656 119 - 01935 251961

* Infix Expression → Postfix Exp.

① (A-B) * (D/E) AB-DE/*

② (A+B↑D)/(E-F)+G ABD↑+EF-/G+

③ A*(B+D)/E-F*(G+H/K)

/BD+*E/FGHK/+*-

* (A+B)*C-(D-E)↑F Find Prefix and Postfix

Prefix : ↑-*+ABC-DEF

Postfix: AB+C*DE--F↑

*** Stack এর ব্যবহার লিখ

① Dynamic Programming এর ক্ষেত্রে ব্যবহৃত হয়

② Infix থেকে Postfix তৈরির কাজে

③ Quick Short algorithm তৈরিতে

④ কোন expression এর Value calculation করার জন্য

* Recursion কি?

Recursion বলতে পুনঃ পুনঃ আবর্তিত হওয়া বুঝায়।

Recursion বিষয়টি Computer Programming এর গুরুত্বপূর্ন বিষয়।

* Recursion প্রক্রিয়ায় 4! এর মান নির্ণয় করঃ

1. 4! = 4·3!

3 = 3·2!

2! = 2·1!

1! = 1·0!

0! = 1

1! = 1·1 = 1

2! = 2·1 = 2

3! = 3·2 = 6

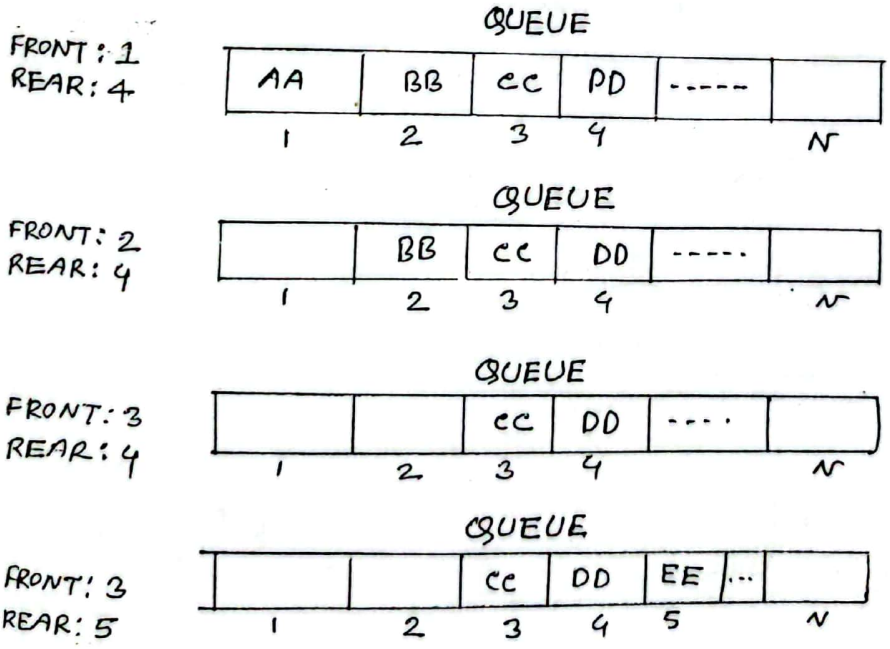
4! = 4·6 = 24

☑ Queue: Queue শব্দের আভিধানিক অর্থ যেমন টিকিটকে- সারিবদ্ধভাবে সাজানো। Queue একটি Linear Data Structure যা FIFO (first-In-first-out) system এ Data Manipulate করে থাকে। Queue এ Data element Insert হবে শেষে। এবং Delete হবে শুরু থেকে।

Basic Operations of Queue:

* enqueue()
* dequeue()
* peek()
* isfull()
* isEmpty()

Array representation of a queue: / Linear Queue Operation



* Classification of QUEUE:

① Linear Queue
② Circular Queue
③ Priority Queue
④ De-Queue

☐ Linear Queue: যে queue তে সাধারণিক নতুন Data Manipulate করার হয় তাকে Linear Queue বলে।

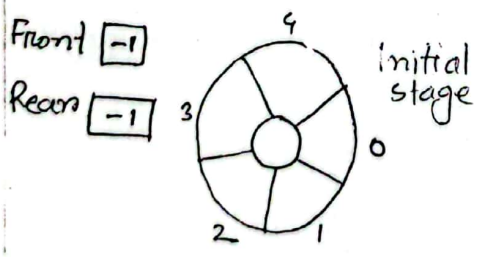
☐ Linear Queue তে Deletion Operation (Dequeue) [Details in class]

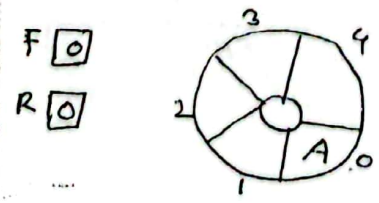
☐ Linear Queue তে Insertion Operation (Enqueue) [Details in class]

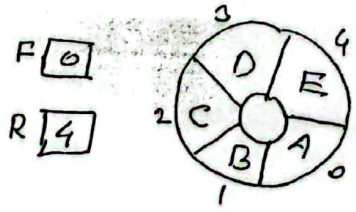
☐ Circular Queue: Linear Queue তে খালি জায়গা বা free space থাকা সত্ত্বেও Data insert করা যায় না। এ সমস্যা সমাধান করার জন্য যে Queue ব্যবহার করা হয় তাহলো Circular Queue.

Circular Queue তে counter ব্যবহার করা হয় যা Queue তে কতগুলো Data আছে তার সংখ্যা নির্দেশ করে। counter এর মান 0 হলে Data থাকে না।

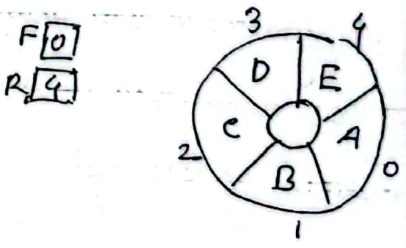
Insertion

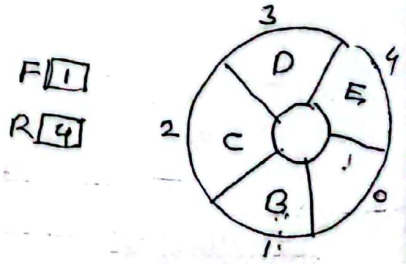


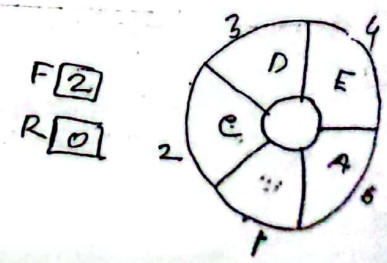




Deletion







☑ Linear Queue এর চেয়ে Circular Queue এর সুবিধা:

① Linear Queue তে Data Insertion, Deletion একবার মাত্রে করা যায় না যা circular queue তে করা যায়

② Linear Queue তে Memory অপচয় হয়, Circular Queue তে হয় না

③ সম্পূর্ন আলি না হওয়া পর্যন্ত Linear Queue তে নতুন কোন রম position এ Data insert করা যায় না

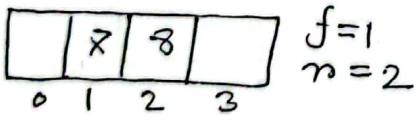
☑ Deque / Double Ended Queue: Deque এমন এক ধরনের queue যেখানে queue এর দুদিক থেকে data insert ও remove করা যায়। কিন্তু queue এর মাঝে insert বা Delete করা যায় না।

Deque বা Double ended queue দুই প্রকার:

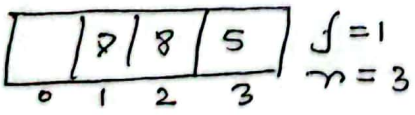
① Input Restricted

② Output Restricted.

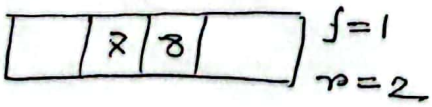
Deque Operation:



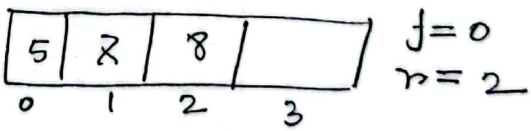
1. AddRear (5)



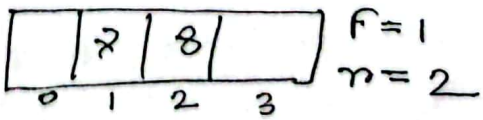
2. DeleteRear()



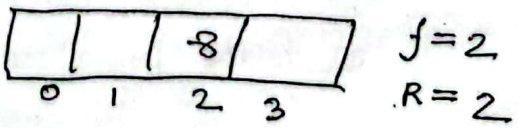
3. Add Front (5)



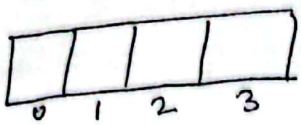
4. Delete Front()



5. Delete Front()



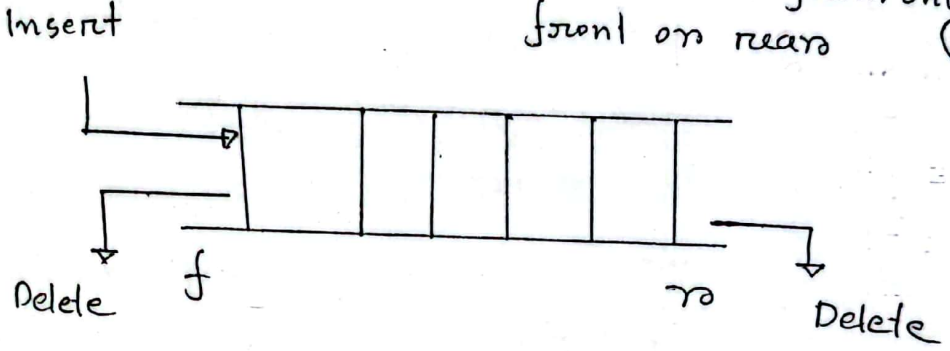
6. Delete Rear ()



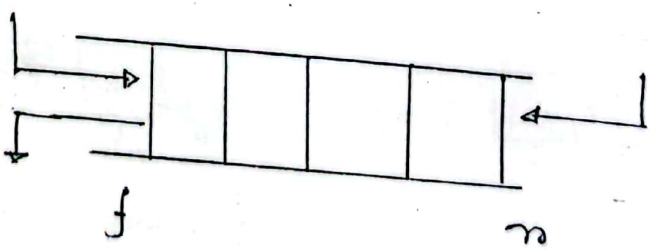
f > r Then we know queue is empty

when f > r and if we want to going initial state of queue then f = 2 → -1 and r = 1 → -1 [Reset Pointers]

Input Restricted Deque: Insertion one from only front or rear



Output Restricted Deque: Deletion only one from and Insertion both from.

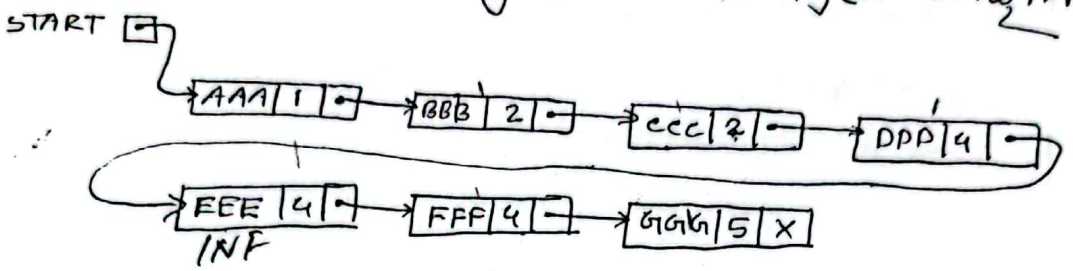


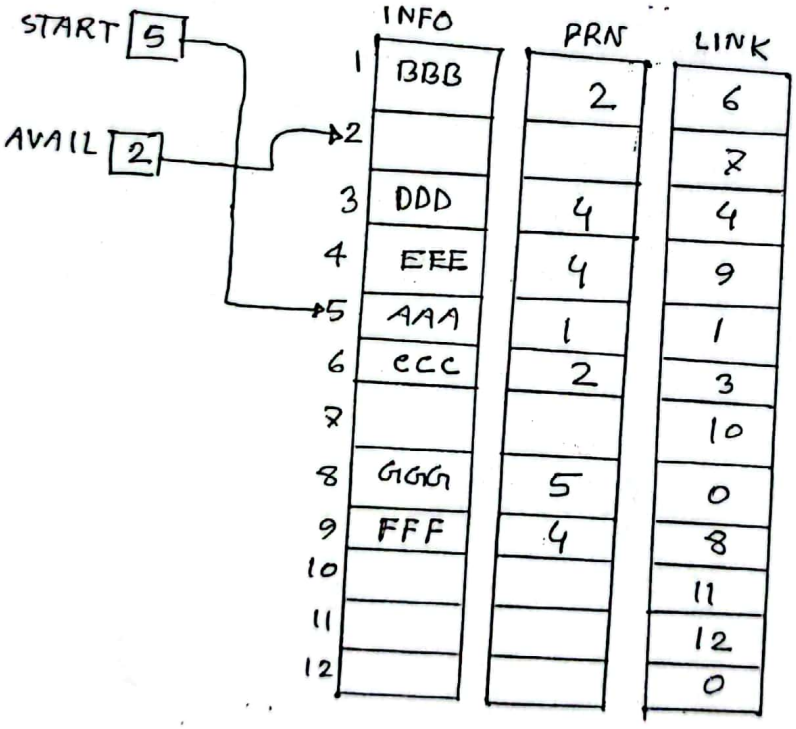
Deque perform 4 operations :—

(i) insert at front
(ii) delete from front
(iii) insert at rear
(iv) delete from rear

☐ Priority Queue: এটি এমন এক ধরনের queue যেখানে প্রত্যেকটি Data element এর priority হিসাবে Manipulate হয়। এক্ষেত্রে যে সংখ্যা সবচেয়ে ছোট (Assending) এর priority সবথেকে বেশি।

** একটি List কে Priority Queue এর মাধ্যমে উপস্থাপন





** Gives few examples for data structures?

- Stacks
- Queue
- Link List
- Trees
- Graphs

Algorithm for ENQUEUE Operation

1. check if the queue is full or not
2. If the queue is full, then print overflow error and exit the program
3. If the queue is not full, the increment the tail and add the element

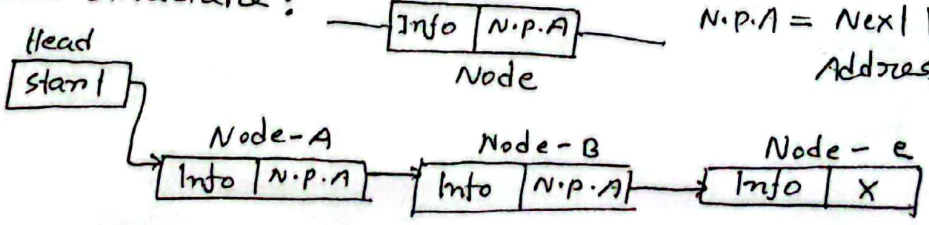
Algorithm for DEQUEUE operations

1. check if the queue is empty or not
2. If the queue is empty, the print underflow error and exit the program
3. If the queue is not empty, then print the element at the head and increment the head.

Linked List : কোন সিস্টেম বা তালিকায় সংরক্ষিত তথ্য সমূহের প্যারামিটারকে যদি ক্যারেক্টার অনুযায়ী সাজানো থাকে তবে ঐ ধরনের সিস্টেমকে সিকুয়েন্স সিস্টেম বলে।
Linked List একটি Dynamic Linear Data structure যা Dynamic Memory Allocation এ ব্যবহৃত হয়।
Linked List এর প্রত্যেকটি cell কে node বলা হয় এবং এর pointing এর মাধ্যমে Data store করে। এতে Head নামক একটি variable ব্যবহৃত হয়।

X = Null Pointer
N.P.A = Next Pointer Address

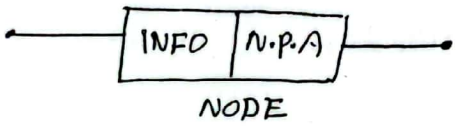
Basic Structure :



Classification of Linked List

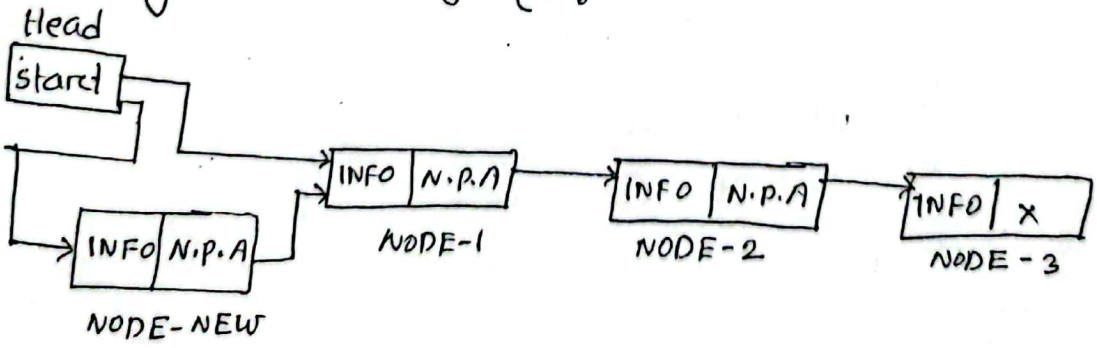
① Linear Linked List
② Header " "
③ Circular " "
④ Double/ Two-way L.L

Linear Link List: Linear L.L এমন একটি one way L.L যার প্রত্যেকটি Node এর দুটি অংশ থাকে। একটি অংশে node এর Information এবং অন্য অংশে- Next pointer Address ধারন করে। এতে Data Manipulation linearly হয় বিধায় একে Linear Linked List বলা হয়।



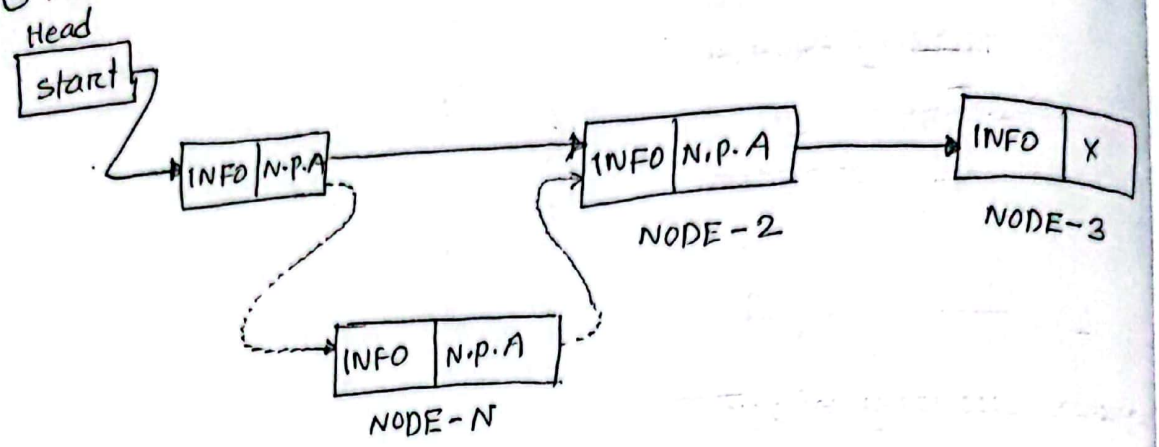
⇒ **Insertion of Linear Linked List:** Linear Linked list এ তিনটি position এ নতুন Node insert করা যায়।
① Starting ② Middle ③ Ending

① Starting এ Node সংযোজন:



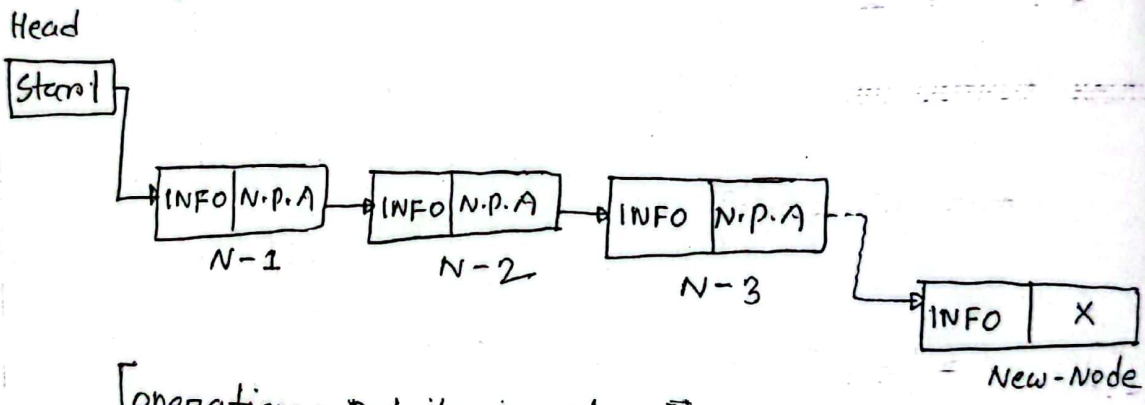
operation: [Details in class]

(ii) Middle এ সংযোজন:



[operation: Details in class]

(iii) Ending এ সংযোজন:

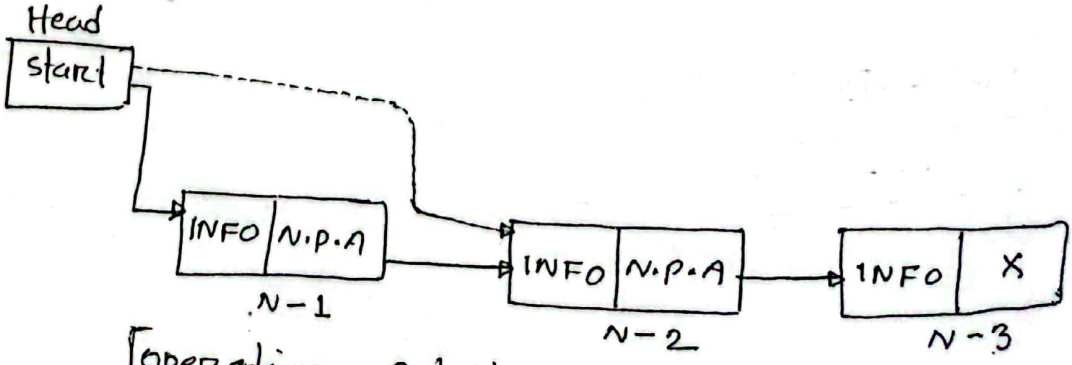


[operations: Details in class]

☑ Deletion of Linear Linked List:

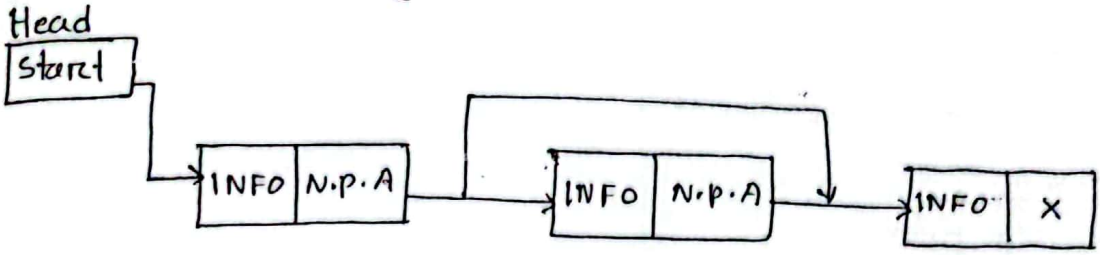
① Starting (ii) Middle (iii) Ending

① Starting - এ বিয়োজন:



[operation: Details in class]

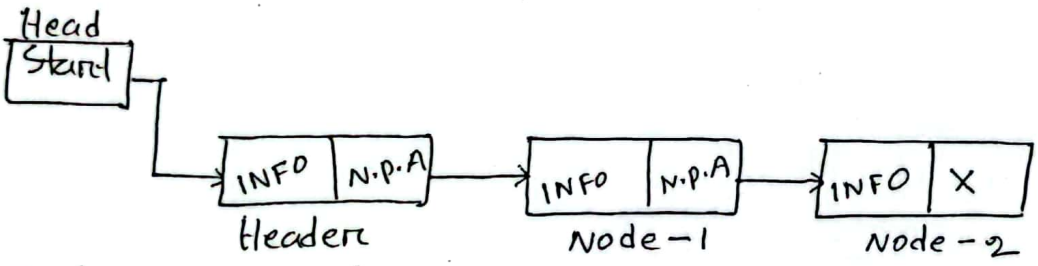
(ii) Middle এ বিয়োজনঃ



[operation : Details in class]

(iii) Ending এ বিয়োজনঃ [Details in class]

☑ Header Linked List: Header Linked List এমন এক-ধরনের -Linked List যাতে Header নামে একটি অতিরিক্ত Node থাকে-।



Header Node এর কাজ:

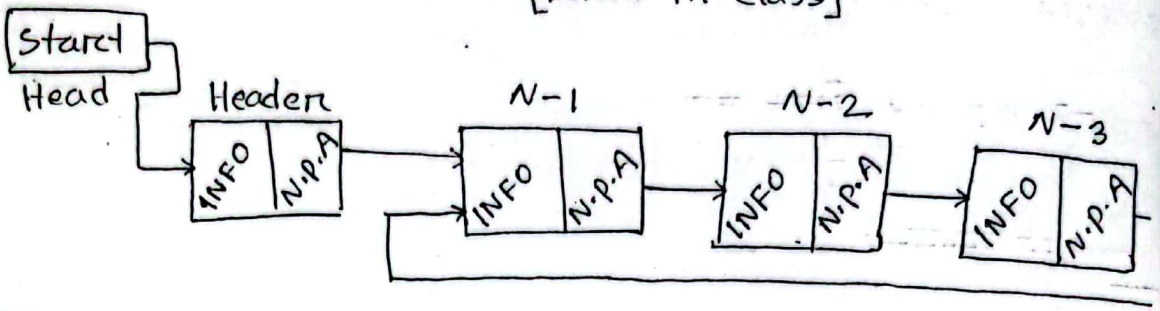
(i) Linked List এর ordered বা এলোমেলো জানা যায়
(ii) Sorting জানা যায়
(iii) Length জানা যায়
(iv) Datatype জানা যায়

Classification of Header Linked List:

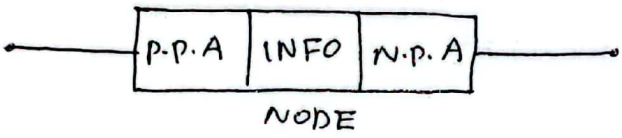
(i) Circular Header
(ii) Two-way / Doublly Header
(iii) Two-way circular

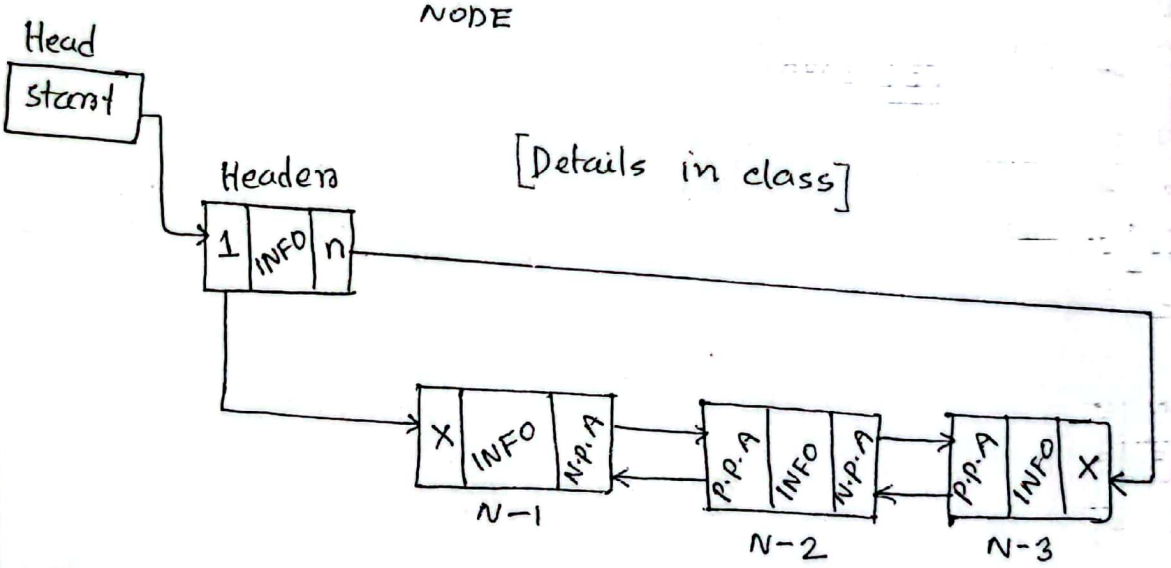
① Circular Header:

[Details in class]

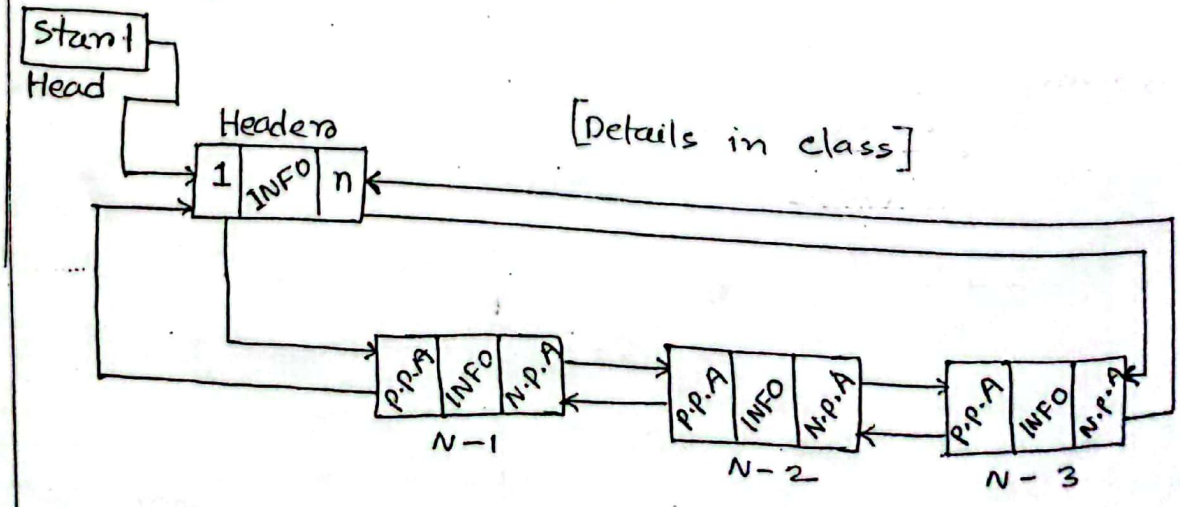


⑪ Two-way Header Linked List:

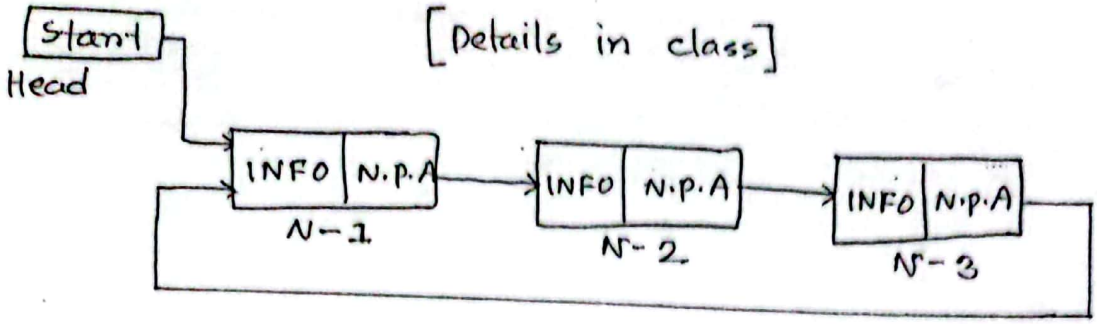




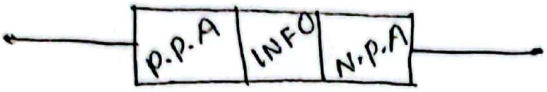
☒ Two-way circular Header:

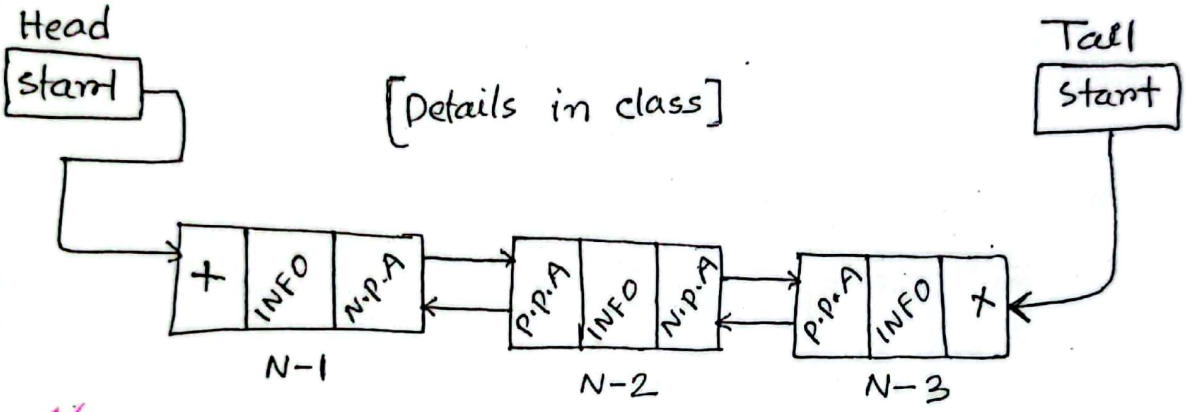


Circular Linked List:

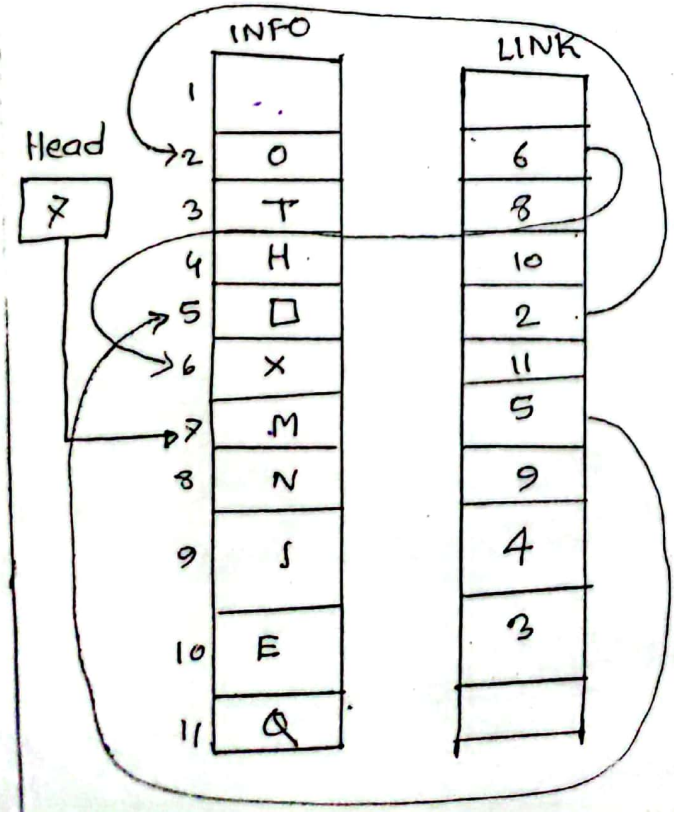


Two-way / Doubly Linked List:



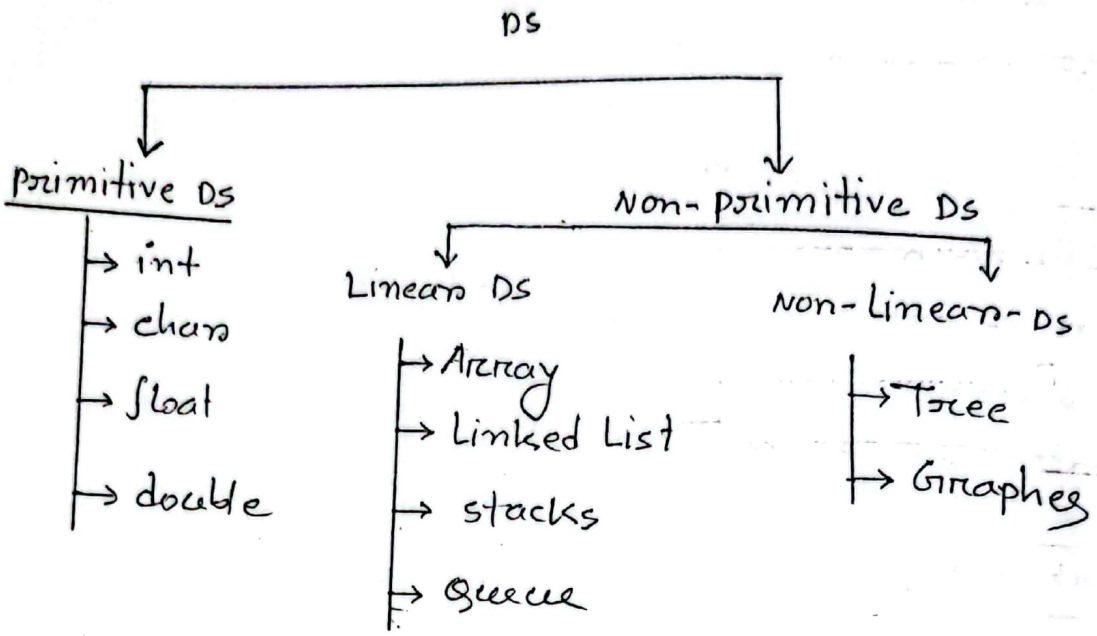


Representation of Linked List in Memory



Start = 7 INFO[7] = M
LINK[7] = 5 INFO[5] = □
LINK[5] = 2 INFO[2] = O
LINK[2] = 6 INFO[6] = X
LINK[6] = 11 INFO[11] Q

☑ Type of Data Structure:



☑ Garbage collection: Memory তে store কৃত একটি list-এর কোন একটি node delete করার ফলে অথবা একটি program থেকে অযথা List Delete করার ফলে memory তে Available space সৃষ্টি হয়। computer এর operating system অযথা delete space গুলোকে periodically সংগ্রহ করে free storage list গঠন করে। এই collection এর Technique কে বলা হয় Garbage collection.

☑ Link List এর সুবিধা:

① L.L এ Data এর size পূর্বে নির্ধারণের প্রয়োজন হয়না

② যে কোন সংখ্যক Data নিয়ে কাজ করা যায়

③ New Data insert করার জন্য অথবেই List এর size বৃদ্ধি করা যায়

④ Memory অপচয় রোধ করে।

PSC- 01935 251961 - 01870-656119 ন্য

☑ সর্টিং (sorting) : সর্টিং বলতে কোন ডাটার উপাদান সমূহকে মানের ক্রমানুসারে সাজানোর পদ্ধতিকে বুঝায়। সংখ্যা বা ক্যারেক্টারকে দুভাবে সাজানো যায় —

(i) Ascending Order
(ii) Descending Order

বিভিন্ন ধরনের সর্টিং -

(i) Bubble Sort
(ii) Quick Sort
(iii) Heap Sort
(iv) Insertion Sort
(v) selection Sort
(vi) Radix Sort
(vii) Merge Sort

[কিছু সর্টিং পদ্ধতি ক্লাসে আলোচনা করব]

☑ সার্চিং (Searching) :

সার্চ শব্দের আভিধানিক অর্থ হলো অন্বেষন করা বা খোঁজ করা। অর্থাৎ একটি উপাদান সমূহ হতে কোন নির্দিষ্ট মানের উপাদানসমূহকে খুজে বের করার প্রক্রিয়াকে সার্চিং বলে।

searching পদ্ধতি সাধারনত 2 প্রকার :

(i) Linear search : একটি লিনিয়ার ডাটা থেকে প্রতিটি উপাদানের সাথে পর্যায়ক্রমিক তুলনা করে কোন নির্দিষ্ট তথ্য বা আইটেমকে খুজে বের করার পদ্ধতিকে Lenear search বলে।

(ii) Binary search : এ পদ্ধতিতে অতি দ্রুত কোন আইটেমকে সার্চ করা যায়। Binary search পদ্ধতি খুব তখনই বা সাজানো Data র ক্ষেত্রে প্রয়োগ করা হয়। সাধারনত ডাটা গুলো উর্ধ্বক্রমানুসারে সাজানো থাকে।

☑ Hashing table : Hashing মূলত একটি সার্চিং কৌশল। Hashing এমন এক ধরনের searching কৌশল যে পদ্ধতিতে সার্চিং এর সময় ডাটা কম্পেয়ারের উপাদান সংখ্যার উপর নির্ভরশীল নয়। Hashing সম্পন্ন করার জন্য Hashing Table গঠন করা যায় যে টেবিলে সংরক্ষিত ডাটার অ্যাড্রেস Hashing Address ব্যবহার করা হয়।

Hashing Table ব্যবহার করার ফলে মেমোরিতে অযথা মেমরি বিনষ্ট হয় না।

☑ Hashing function : যে function ব্যবহার করে প্রচলিত টেবিলের মেমোরি অ্যাড্রেস পরিবর্তন করে হ্যাশিং এড্রেস নির্ধারণ করা হয় তাকে Hashing function বলে।

☑ Algorithm: কোন সমস্যা সমাধানের সুস্পষ্ট ধাপসমূহের ক্রমবিন্যাসকে Algorithm বলে।

☑ সিউডো কোড (pseudo code):
কোন সমস্যা সমাধানের সুস্পষ্ট ধাপসমূহকে যখন প্রোগ্রামিং ভাষায় নির্দেশ অনুসারে ইংরেজী ভাষার সংক্ষিপ্ত শব্দের মাধ্যমে লেখা হয়ে থাকে তখন তাকে pseudo code বলে একে programming Design Language বা সংক্ষেপে PDL ও বলে।