DATA STRUCTURES
ACCS – 16301

Assignment -1

Total Marks :24

Student Name : Sumit Kumar Giri

Class CSE **Sem** 3rd **Univ RollNo.** 2000213 (CS2)

Section – A **(6 Ques * 2Marks=12 Marks)**

Question 1:

Ques A: What is data, fields, records, information. Differentiate between data and Information

Answer : The term data simply refers to a value which may represent some observation from an experiment gathered systematically for one or more specific purpose.
Fields:- A collection of data is frequently organized into field.
Records:- A file is a collection of related record. INFORMATION :- Information is defined as the processed, summarized data which when used by its recipient, helps in taking decision. D/B data and infor:- (i) Data is in the form of numbers, letter etc. but information is the form of Idea. (ii) Data measured in bits, bytes but information is time etc.

Ques B : Define term Time Space Complexity.

Answer : The analysis of algorithms based on time of computation is called time space complexity of an algorithm. The main objective of time complexity is to compare the performance of different algorithms in solving the same problem.

Ques C: Define best case, average case and worst case for an algorithm, Also Define Big O Notation.

Answer : Best case:- It is the shortest running time of an algorithm for all input of given size. Average Case:- It is the average running time for all input of a given size. worst case:- It is the longest (worst) running time of an algorithm for all input of given size.
Big O Notation:- Big-O is standard mathematical notation that shows how efficient an algorithm in the worst-case scenario relative to its input size.

Ques D: Imagine an array A(2:6,-3:8,4:8). Calculate total no. of elements

Answer : No. of elements for 3D Array $L_1 * L_2 * L_3$

$L_1 = U.B - L.B + 1$
$L_1 = 6 - 2 + 1 = 5$
$L_2 = U.B_2 - L.B_2 + 1$
$L_2 = 8 - (-3) + 1 = 12$
$L_3 = U.B_3 - L.B_3 + 1$
$L_3 = 8 - 4 + 1 = 5$

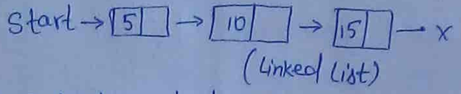
No. of element $= 5 * 12 * 5 = 300$

Ques E: Explain the term Garbage collection. Which component of a PC performs it ?

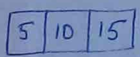
Answer : Garbage collection is the systematic recovery of pooled Computer storage that is Being used by a program when that program no longer needs the storage. There are many Component of a PC performs.
(i) motherboard (ii) Processor (iii) Memory (RAM) (iv) powe Supply (v) Hard disk
(vi) Keyboard (vii) mouse (viii) Monitor (ix) Sound Card (x) Control (xi) Input/output

Ques F: Write algorithm for copying array elements to a linked list.

Answer :



(Linked List)



Array

```
(i) Set ptr = start
(ii) ptr(data) = arr[0]
     Ptr(Next) = NULL
(iii) Initialise i = 1, till i<n
     Ptr(next) = ptr 2;
     ptr2(data) = arr(i)
     2(Next) = null
     ptr = Ptr2
Increment (i) by 1
```

Section – B (3 Ques * 4Marks=12 Marks)

Ques 2: Write Binary search technique with its algorithm .

Answer : The binary search algorithm applied to our array data works as follow. During each stage of our algorithm our seach for ITEM is reduced to a segment of element of data.

Data[BEG], DATA[BEG+1], DATA[BEG+2], ----, DATA[END]

The algorithm Compares item with the middle element DATA[MID] of the sequence, where MID is obtained by

MID = INT((BEG+END)/2)

we use INT(A) for the integer value A. If DATA[MID] = ITEM then the search is successful and we set LOC = MID. otherwise a new segment of DATA is obtained as follow.

(a) If ITEM < DATA[MID], then Item Can appear only in the left hold of the segment.

DATA[BEG], DATA[BEG+1], --- DATA [MID-1]

(b) If ITEM > DATA[MID], then ITEM can appear only in the right hold of the segment.

DATA[MID+1], DATA[MID+2] ...... DATA[END]

So reset BEG = MID+1 and begin searching again. If ITEM is not in DATA, then eventually we obtain

EN < BEG.

Algorithm;

```
(i) [Initialize segment variables]
Set BEG = LB, END = UB and MID = INT (BEG + END)/2
(ii) Repeat steps 3 and 4 while BEG ≤ END and DATA[MID] ≠ 1
(iii) If item < DATA[MID], Keni.
        Set END = MID-1
    else
        Set BEG = MID+1
    [End of if structure]
(iv) Set MID = INT[BEG + END/2)
    [End of step 2 loop]
(v) If DATA[MID] = ITEM, then:
        Set LOC = MID
    else
        Set LOC = NULL
    [End of if structure]
(vi) Exit
```

Ques 3: Imagine a 3D Array A(2:6,3:8,1:6), Base(A)=200, w=4, Calculate Loc(A[4,4,2])

Answer : Given :

Base (A) = 200 , W = 4

LOC(A[4,4,2]) , A (2:6, 3:8, 1:6)

So,

$K_1 = 4$	$L.B_1 = 2$	$U.B_1 = 6$
$K_2 = 4$	$L.B_2 = 3$	$U.B_2 = 8$
$K_3 = 2$	$LB_3 = 1$	$U.B_3 = 6$

$A(5,6,6)$

$L_1 = 5$

$L_2 = 6$

$L_3 = 6$

$\therefore E_1 = K_1 - LB_1$

$= 4 - 2 = 2$

$E_2 = K_2 - LB_2$

$= 4 - 3 = 1$

$E_3 = K_3 - LB_3$

$E_3 = 2 - 1 = 1$

$CMO = Base(A) + w[(E_2L_2 + E_2)L_1 + E_1]$

$= 200 + 4[(1\times6+1)5+2]$

$= 200 + 4 \times 37$

$= 200 + 148$

$\boxed{CMO = 348}$

$RMO = Base(A) + w[(E_1L_2 + E_2)L_1 + E_1]$

$= 200 + 4[(2\times6+1)6+1]$

$= 200 + 4[79]$

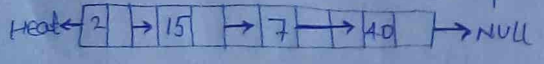
$= 200 + 316$

$\therefore \boxed{RMO = 516}$

Ques 4: Explain all Linked list Insertion and deletion cases with diagram for each.

Answer : Linked list is a data structure used for storing collection of data. A linked list has following properties:

- Successive elements are connected by pointer.
- The lost element point to Null.
- Can grow.
- Can be made just as long as requred.
- Does not woste memory space.



- Main Linked List operation.
  - Insert: insert an element into the list
  - Delete: Remove and relum the specified element from the list.

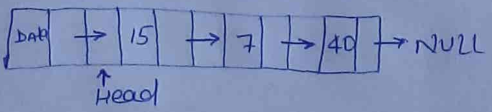
(i) Singly linked list:-

Generally, 'linked list' means a singly linked list. This list Consist of a number of notes in which each note in which each note has a next pointer of the following element.

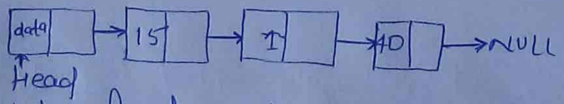
Insertion into a singly-Linked list has three Cases:

- Inserting a Blew node before the head (at the beging)
- Inserting a new node after the fail. (at thread of the list)
- Inserting a new node at the middle of the list.

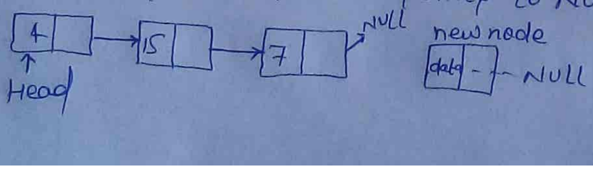
- Inserting a node is singly Linked list at the Beginning in this case, a new node is inserted before the Current head node, only one next pointer needs to be modified and it Can be done in two steps.



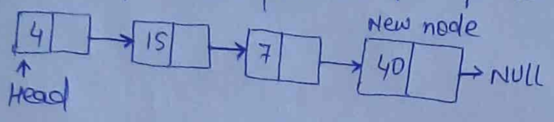
- update is the next Pointer of new node, to point to the Current head.



- update head pointer to point to the new node. Inserting ~~case~~ a Node is singly linked list at the ending In this case, we need to modify the next pointer.
- New nodes next pointer points to NULL.



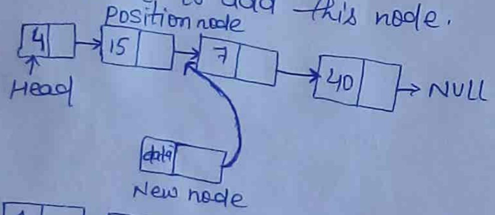
- Last nodes next pointer points to the new nodes.

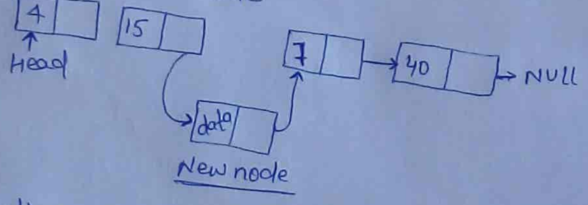


Inserting a node in singly Linked list at the middle.

Let us assume that we are given a position where we want to insert the new node. In this Case, we need to modify two next pointers.

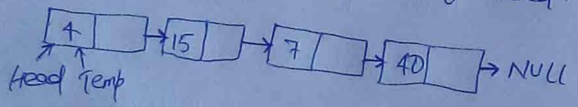
- If we want to add an element at position 3 then we stop at position 2. that means we traverse two nodes and insert the new node for simplicity Let us assume that the second node is called position node the new node points to the next node of the position where we want to add this node.



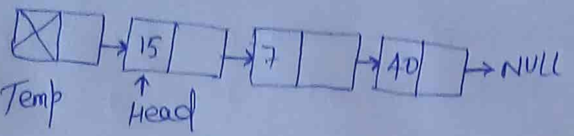


Deletion

- Deleting the first Node is singly linked list. It can be done in two steps.
  - Creat a temporary node which will point to be Same node as that of head.



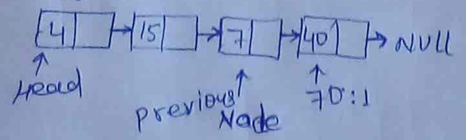
- Now, move the head notes pointer to the next node and dispose of the tempory node.



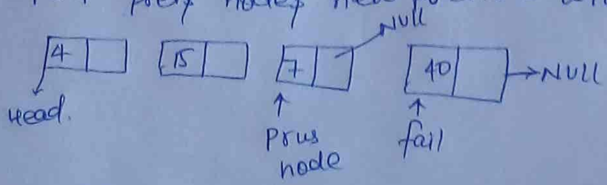
Deleting the last Node in singly linked list.

In this Case, the last node is removed from the list, this operation is a bit trickier than removing the first node, becouse the algorithm should find a node, which is prus to the fail. It Can be done in three steps.

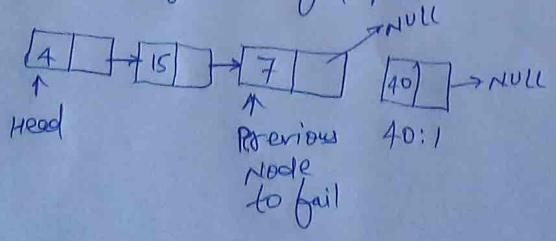
- Traverse the list and while traversing maintain the previous node address also. By the time we reach the end of the list, we will have two pointers, one pointing to the fails node and the other opinting to the node before the fail node.



- Updat prevs nodes new poerious with NULL.



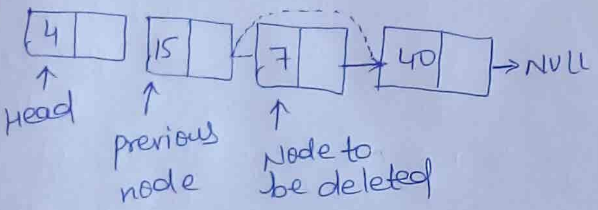
- Dispose of the fail Node.



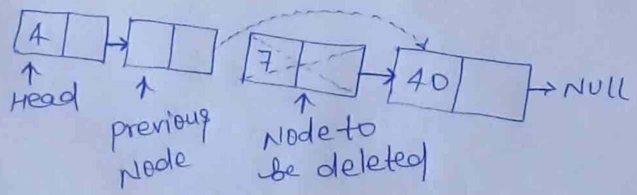
Deleting an Intermediate Node is singly Linked list:

In this Case, the node to de ~~bea~~ removed is always locald between two nodes. Head and fail links are not updated in this Case. Such a removal Can be done in two steps:

- Similar to prus Case maintain the previous node write traversing the list, once we find the node to be deleted, Change the previous node next pointer to the next pointer of the node to be deleted.



- Dispase of the Current node to be deleted.


4
7
40
→NULL
Head
previous Node
Node to be deleted