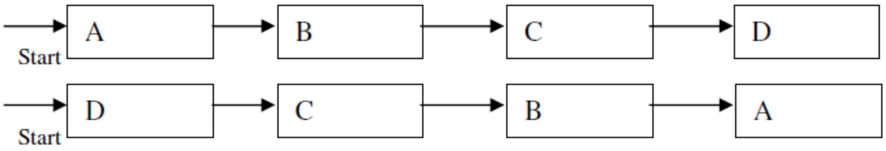
USN □□□□□□□□□□

Internal Assessment Test 2 – Dec 2022

Sub:	Data Structures and Applications					Sub Code:	21CS32	Branch: CSE
Date:	27 /12 /2022	Duration:	90 mins	Max Marks:	50	Sem / Sec:	III(A,B & C)	OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 (a)	Consider the following sequence of operations on an empty stack. push(54); push(52); pop(); push(55); push(62); s = pop(); Consider the following sequence of operations on an empty queue. enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32); q = dequeue(). Demonstrate the above sequence of operation on a stack and queue with a help of a neat diagram and predict the value of s + q	[05]	CO2	L3
(b)	Write a note on Dequeue and Priority Queues.	[05]	CO2	L2
2	List the advantages of circular queue over ordinary queue? With suitable C-functions simulate the working of circular Queue of integers using Arrays. Suppose a queue is maintained by a circular array queue with N=12 memory cells. Find the number of elements in the queue if i) FRONT =4 REAR =8 ii) FRONT =10 REAR = 3 iii) FRONT =5 REAR =6 and then two elements are deleted.	[10]	CO2	L3
3	Write C functions to perform the following operations in a SLL: i) Assume a four node single linked list with data values 15,25,40,50 ii) Insert a node with data value ‘60’ at the end of the list. iii) Insert a node with data value 30 in between the nodes 25 and 40 iv) Delete a node with data value ‘40’ v) Search node with data value ‘25’	[10]	CO1	L3
4	Write C functions to perform the following operations in the SLL in figure below: i. To count number of nodes in the given singly linked list. ii. To reverse direction of singly linked list (as shown below). iii. To concatenate the two singly linked list. 	[10]	CO3	L3
5	Describe the doubly linked list with advantages and disadvantages. Write necessary C- functions to perform the following: iv. Insert a node at the front of DLL v. Delete a node from the front of DLL vi. Insert a node from a DLL before a node with a given value. vii. Delete a node from a DLL before a node with a given value.	[10]	CO3	L2
6	Demonstrate the various operations performed in Linked Queue with suitable C-function.	[10]	CO3	L2

CI **CCI** **HOD**

PO Mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1																		
CO2																		
CO3																		
CO4																		
CO5																		

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				

Internal Assessment Test 2 – Dec 2022 (Solution)

Sub:	Data Structures and Applications	Sub Code:	21CS32	Branch:	CSE

Solution

Answer any FIVE FULL Questions — MARKS | CO | RBT

1 (a) Consider the following sequence of operations on an empty stack. push(54); push(52); pop(); push(55); push(62); s = pop(); Consider the following sequence of operations on an empty queue. enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32); q = dequeue().
Demonstrate the above sequence of operation on a stack and queue with a help of a neat diagram and predict the value of s + q

Push(54)	**Push(52)**	**Pop()**	**Push(55)**	**Push(62)**	**s=Pop()**
				62	
	52		55	55	55
54	54	54	54	54	54

EQ(21)	21			
EQ(24)	21	24		
DQ()		24		
EQ(28)		24	28	
EQ(32)		24	28	32
DQ()=24			28	32

Answer = 86

[05] CO2 L3

(b) Write a note on Dequeue and Priority Queues.

The deque stands for Double Ended Queue. Deque is a linear data structure where the insertion and deletion operations are performed from both ends. We can say that deque is a generalized version of the queue.

Though the insertion and deletion in a deque can be performed on both ends, it does not follow the FIFO rule. The representation of a deque is given as follows -

Types of deque

There are two types of deque -

- Input restricted queue
- Output restricted queue

Input restricted Queue

In input restricted queue, insertion operation can be performed at only one end,

[05] CO2 L2

while deletion can be performed from both ends.

Output restricted Queue

In output restricted queue, deletion operation can be performed at only one end, while insertion can be performed from both ends.

A **priority queue** is a type of queue that arranges elements based on their priority values. Elements with higher priority values are typically retrieved before elements with lower priority values.

In a priority queue, each element has a priority value associated with it. When you add an element to the queue, it is inserted in a position based on its priority value. For example, if you add an element with a high priority value to a priority queue, it may be inserted near the front of the queue, while an element with a low priority value may be inserted near the back.

There are several ways to implement a priority queue, including using an array, linked list, heap, or binary search tree. Each method has its own advantages and disadvantages, and the best choice will depend on the specific needs of your application.

2 List the advantages of circular queue over ordinary queue? With suitable C-functions simulate the working of circular Queue of integers using Arrays.

Suppose a queue is maintained by a circular array queue with N=12 memory cells. Find the number of elements in the queue if

i) FRONT =4 REAR =8

ii) FRONT =10 REAR = 3

iii) FRONT =5 REAR =6 and then two elements are deleted.

Q-1

0	1	2	3	4	5	6	7	8	9	10	11
				F				R			

Q-2

0	1	2	3	4	5	6	7	8	9	10	11
			R							F	

Q-3

0	1	2	3	4	5	6	7	8	9	10	11
					F	R					

[10] CO2 L3

3 Write C functions to perform the following operations in a SLL:

i) Assume a four node single linked list with data values 15,25,40,50

ii) Insert a node with data value ‘60’ at the end of the list.

[10] CO1 L3

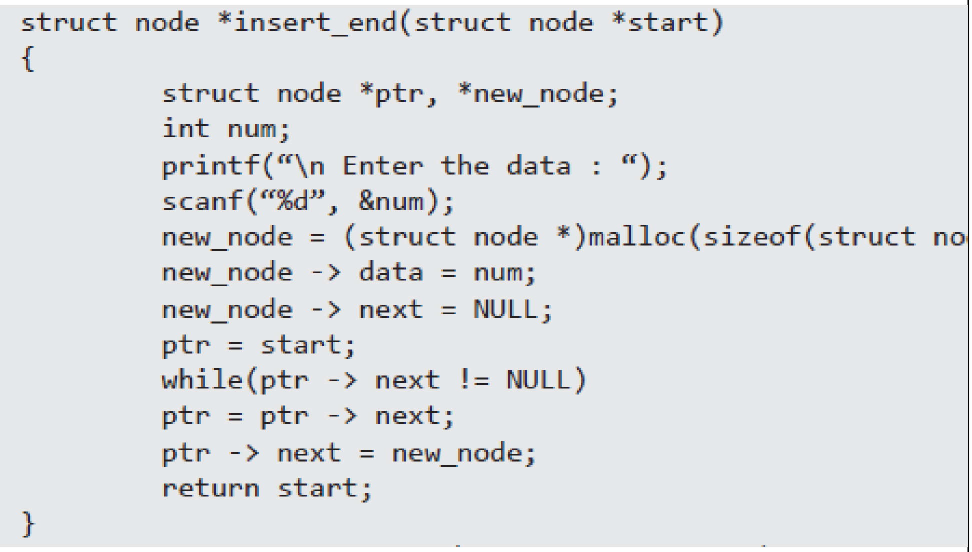
iii) Insert a node with data value 30 in between the nodes 25 and 40

iv) Delete a node with data value ‘40’

v) Search node with data value ‘25’

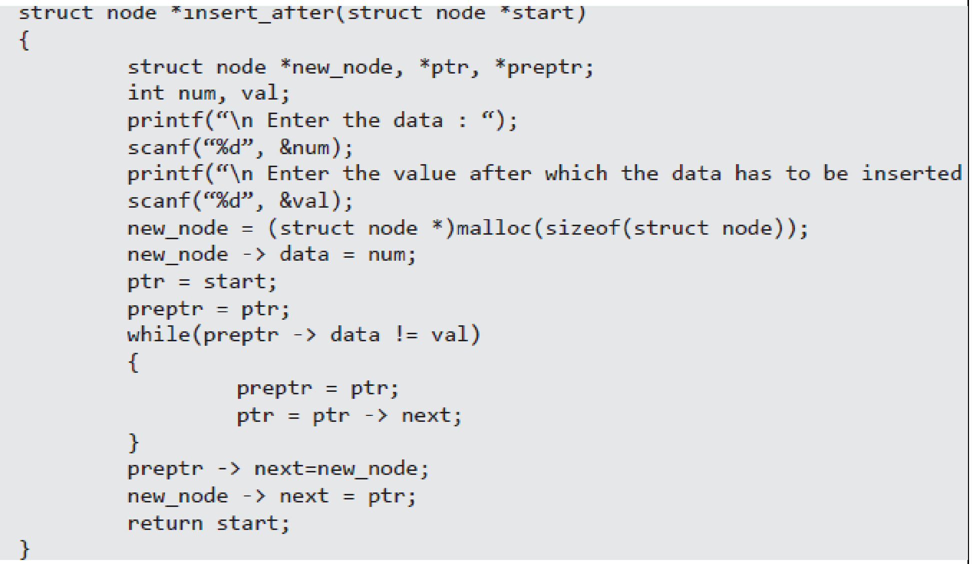
i) Assume a four node single linked list with data values 15,25,40,50

ii) Insert a node with data value ‘60’ at the end of the list.



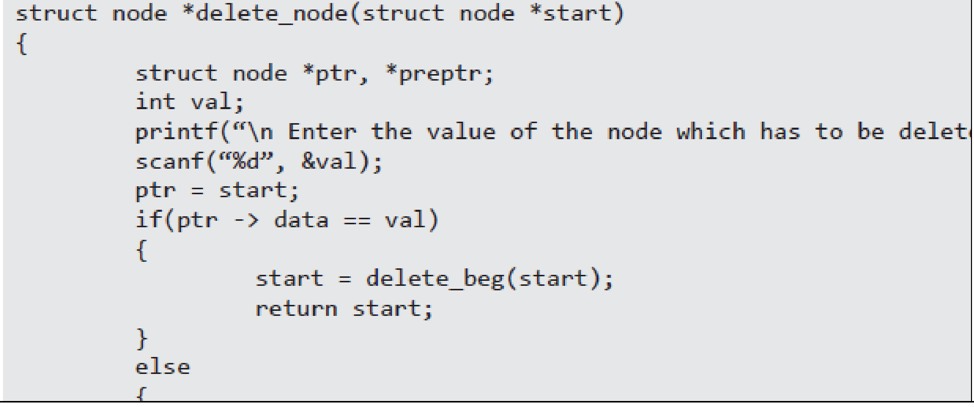
```
struct node *insert_end(struct node *start)
{
        struct node *ptr, *new_node;
        int num;
        printf(“\n Enter the data : “);
        scanf(“%d”, &num);
        new_node = (struct node *)malloc(sizeof(struct no
        new_node -> data = num;
        new_node -> next = NULL;
        ptr = start;
        while(ptr -> next != NULL)
        ptr = ptr -> next;
        ptr -> next = new_node;
        return start;
}
```

iii) Insert a node with data value 30 in between the nodes 25 and 40

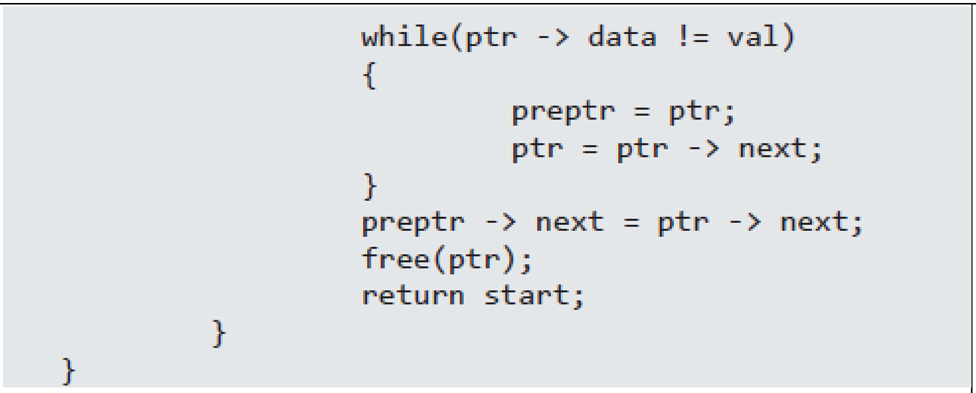


```
struct node *insert_after(struct node *start)
{
        struct node *new_node, *ptr, *preptr;
        int num, val;
        printf(“\n Enter the data : “);
        scanf(“%d”, &num);
        printf(“\n Enter the value after which the data has to be inserted
        scanf(“%d”, &val);
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
        ptr = start;
        preptr = ptr;
        while(preptr -> data != val)
        {
                preptr = ptr;
                ptr = ptr -> next;
        }
        preptr -> next=new_node;
        new_node -> next = ptr;
        return start;
}
```

iv) Delete a node with data value ‘40’



```
struct node *delete_node(struct node *start)
{
        struct node *ptr, *preptr;
        int val;
        printf(“\n Enter the value of the node which has to be delet
        scanf(“%d”, &val);
        ptr = start;
        if(ptr -> data == val)
        {
                start = delete_beg(start);
                return start;
        }
        else
        {
```



```
                        while(ptr -> data != val)
                        {
                                preptr = ptr;
                                ptr = ptr -> next;
                        }
                        preptr -> next = ptr -> next;
                        free(ptr);
                        return start;
                }
        }
```

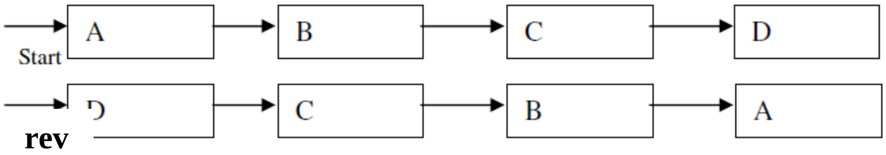
v) Search node with data value ‘25’

4 Write C functions to perform the following operations in the SLL in figure below:

i. To count number of nodes in the given singly linked list.

ii. To reverse direction of singly linked list (as shown below).

iii. To concatenate the two singly linked list.



Count:

```
void print(){
/* temp pointer points to head */
struct node* temp = head;
/* Initialize count variable  */
int count=0;
/* Traverse the linked list and maintain the count */
while(temp != NULL){
temp = temp->next;
/* Increment count variable. */
count++;
}
/* Print the total count. */
printf("\n Total no. of nodes is %d",count);
```

[10] CO3 L3

Reverse:

```
static void reverse(struct Node** head_ref)
{
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* next = NULL;
    while (current != NULL) {
        // Store next
        next = current->next;

        // Reverse current node's pointer
        current->next = prev;

        // Move pointers one position ahead.
        prev = current;
        current = next;
    }
    *head_ref = prev;
}
```

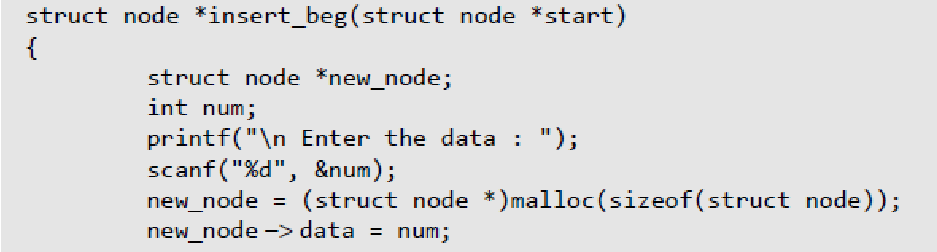
Concatenate:

```
void Concat(struct Node *first, struct Node *second)
{
   struct Node *p = first;
   while (p->next != NULL)
   {
      p = p->next;
   }
   p->next = second;
   second = NULL;
}
```

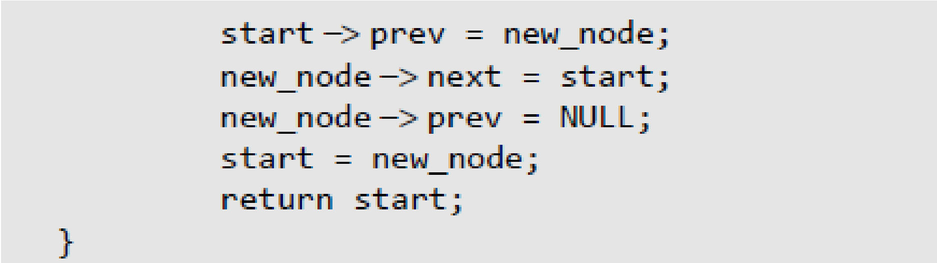
5 Describe the doubly linked list with advantages and disadvantages. Write necessary C- functions to perform the following:

i. Insert a node at the front of DLL
ii. Delete a node from the front of DLL
iii. Insert a node from a DLL before a node with a given value.
iv. Delete a node from a DLL before a node with a given value.

i. Insert a node at the front of DLL

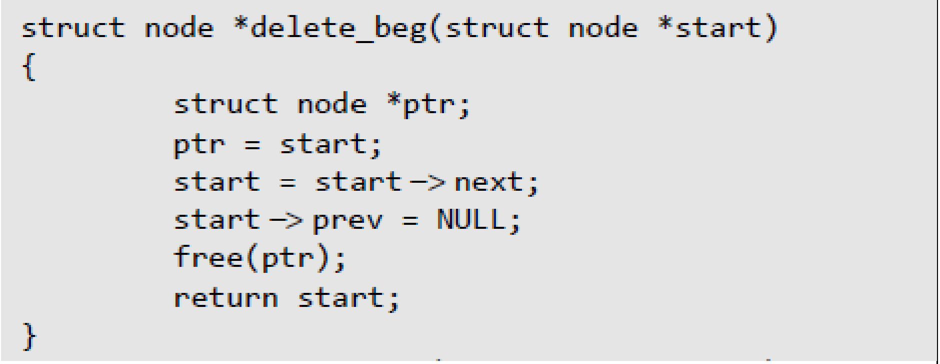


```
struct node *insert_beg(struct node *start)
{
        struct node *new_node;
        int num;
        printf("\n Enter the data : ");
        scanf("%d", &num);
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
```



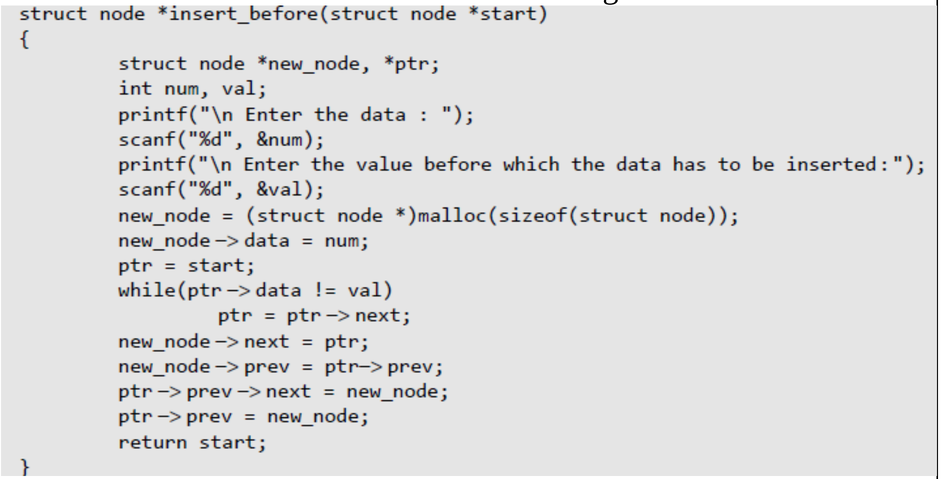
```
        start -> prev = new_node;
        new_node -> next = start;
        new_node -> prev = NULL;
        start = new_node;
        return start;
  }
```

ii. Delete a node from the front of DLL



```
struct node *delete_beg(struct node *start)
{
        struct node *ptr;
        ptr = start;
        start = start -> next;
        start -> prev = NULL;
        free(ptr);
        return start;
}
```

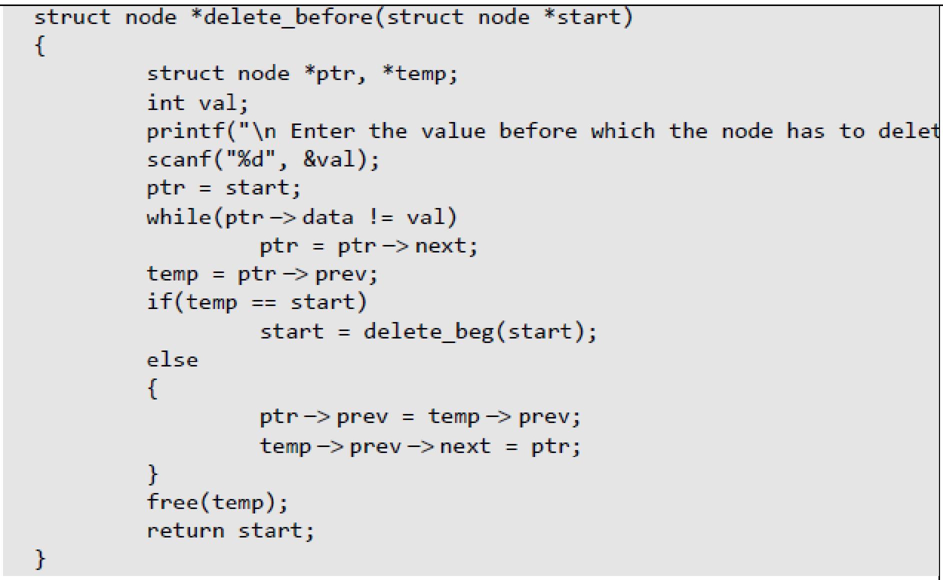
iii. Insert a node from a DLL before a node with a given value.



```
struct node *insert_before(struct node *start)
{
        struct node *new_node, *ptr;
        int num, val;
        printf("\n Enter the data : ");
        scanf("%d", &num);
        printf("\n Enter the value before which the data has to be inserted:");
        scanf("%d", &val);
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
        ptr = start;
        while(ptr -> data != val)
                ptr = ptr -> next;
        new_node -> next = ptr;
        new_node -> prev = ptr-> prev;
        ptr -> prev -> next = new_node;
        ptr -> prev = new_node;
        return start;
}
```

iv. Delete a node from a DLL before a node with a given value.

[10] CO3 L2



```
struct node *delete_before(struct node *start)
{
        struct node *ptr, *temp;
        int val;
        printf("\n Enter the value before which the node has to delet
        scanf("%d", &val);
        ptr = start;
        while(ptr->data != val)
                ptr = ptr->next;
        temp = ptr->prev;
        if(temp == start)
                start = delete_beg(start);
        else
        {
                ptr->prev = temp->prev;
                temp->prev->next = ptr;
        }
        free(temp);
        return start;
}
```

6 Demonstrate the various operations performed in a Linked Stack with suitable C-function.

Operations on stack:

Push:

· To insert an item to a stack.

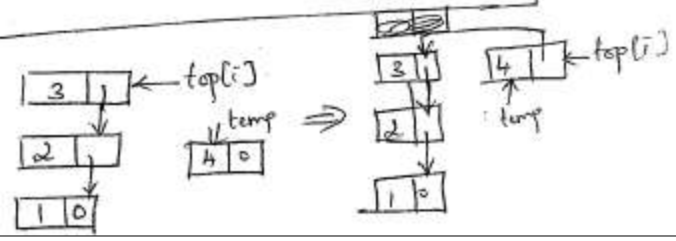
* Create a new node, temp using malloc function.

* Place ~~data~~ item in the data field and top in the link field

* top is then made to point to temp.

```
void push(int i , element item)
 {
   temp= (stackptr) malloc( ~~sd~~ sizeof (stackptr));
   temp->data = item;
   temp->link = top[i];
   top[i]= itemp;
 }
```

Figure:

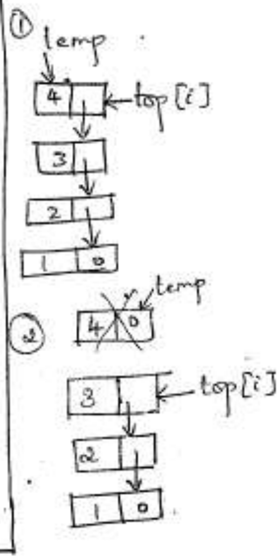


[10] CO3 L2

Pop:

- Pop returns the top element and changes top to point the address contained in its link field.
- The removed node is then freed and item is returned.

```
element pop (int i)
{
   stack ptr  temp = top[i];
   element  item;
   if (! temp)
        return stackEmpty();
   item = temp →data;
   top[i] = temp →link;
   free (temp);
   return item;
}
```


①
temp
4
top[i]
3
2
1
0
②
temp
4
0
3
top[i]
2
1
0