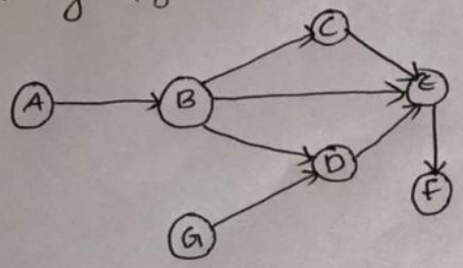
FATHIMA NOVRIN
Data Structure
1st Sem
MCA-217

1. Consider a directed acyclic graph G given in following figure.



Algorithm

Step 1 : write the indegree of all vertices

Step 2 : if there is any vertex have not indegree 0 then there is a cycle and doesit have a topological ordering.

Step 3 : check the vertex have indegree = 0 for the topological sorting

Step 4 : Vertex A has indegree = 0, remove vertex A and its associated edges and update the indegree of the other vertex.

Step 5 : Continue same steps until the graph became empty.

PROGRAM

```
#include<stdio.h>
int main(){
int i,j,k,n,ar[10][10],indeg[10],flag[10],count=0;
printf("enter number of vertices");
scanf("%d",&n);
printf("enter the adjacency matrix");
for(i=0;i<n;i++){
        printf("Enter row %d\n",i+1);
        for(j=0;j<n;j++)
            scanf("%d",&ar[i][j]);
    }

    for(i=0;i<n;i++){
        indeg[i]=0;
        flag[i]=0;
    }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            indeg[i]=indeg[i]+ar[j][i];

    printf("\nThe topological order is:");

    while(count<n){
        for(k=0;k<n;k++){
            if((indeg[k]==0) && (flag[k]==0)){
                printf("%d ",(k+1));
                flag [k]=1;
            }

            for(i=0;i<n;i++){
                if(ar[i][k]==1)
                    indeg[k]--;
            }
        }

        count++;
    }
```

```
    return 0;
}
```

OUTPUT

```
enter number of vertices7
enter the adjacency matrixEnter row 1
0 1 0 0 0 0 0
Enter row 2
0 0 1 1 1 0 0
Enter row 3
0 0 0 0 1 0 0
Enter row 4
0 0 0 0 1 0 0
Enter row 5
0 0 0 0 0 1 0
Enter row 6
0 0 0 0 0 0 0
Enter row 7
0 0 0 1 0 0 0

The topological order is:1 7 2 3 4 5 6
```

2. Write a Program for creating Doubly LL and perform the following operation.

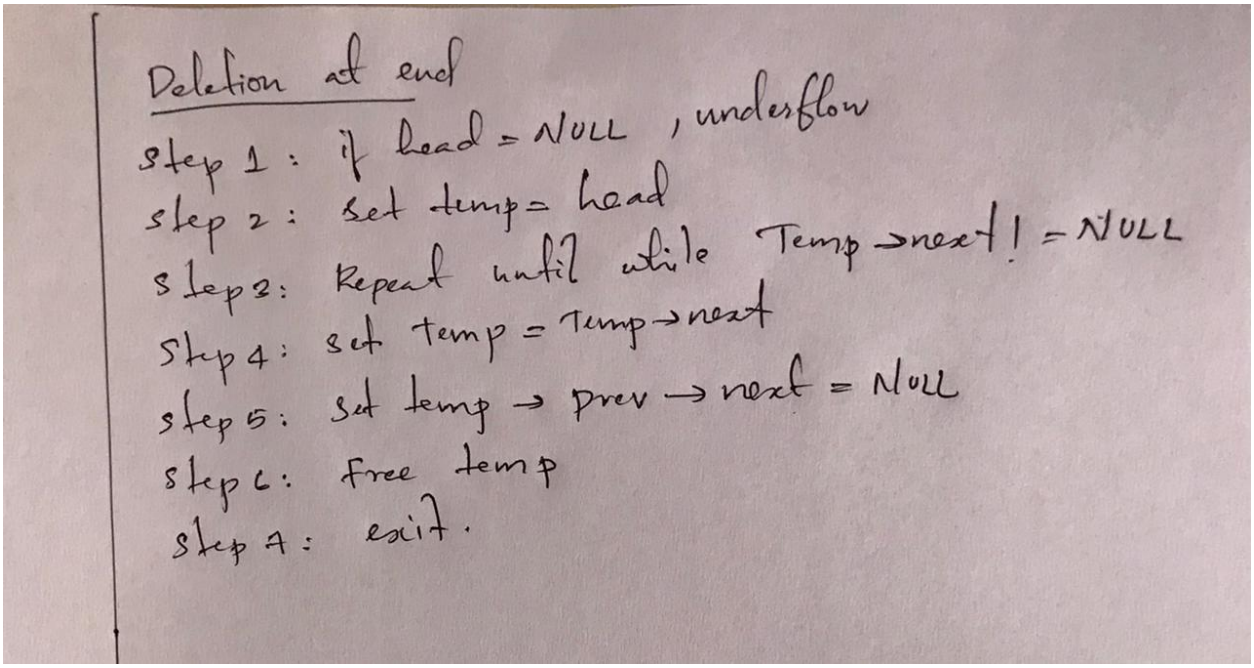
A) insert an element at particular position
B) search an element
c) Delete an element at the end of the list.

Algorithm

A : step 1 : if PTR = NULL , overflow
step 2 : SET New_NODE = PTR
step 3 : SET PTR = PTR → NEXT
step 4 : SET TEMP = START
step 5 : SET I = 0 repeat until I
step 7 : set temp = temp → next
step 8 : if temp = NULL go to step 15
step 9 : set new-node → Next = Temp → next
step 10 : set new_node → prev = temp
step 11 : temp → next = new-node
step 12 : temp → next → prev = new-node
step 15 : exit

Search

step 1 : if head == NULL , underflow
step 2 : set PTR = head , set i = 0
step 3 : Repeat until while PTR != NULL
step 4 : if PTR → data = item , return i
step 5 : i = i+1
step 6 : PTR = PTR → next
step 7 : exit.



PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        struct node *prev;
        struct node *next;
        int data;
};

struct node *head,*last;

void createList(int n);
```

```
void insert_at_anyposition(int data,int position);
void search();
void delete_at_end();
void display();


void main ()
{
	int choice=0,n,data;
	while(choice!=6)
	{
		printf("-----DOUBLY LINKED LIST-----");
		printf("\n1.Create the list \n2.Insert at particular position \n3.Search an element \n4.Delete an element at end \n5.Display \n6.Exit\n");
		printf("Enter your choice : ");
		scanf("\n%d",&choice);

		switch(choice)
		{
			case 1:printf("Enter number of nodes to be inserted: ");
				scanf("%d",&n);
				createList(n);
				break;
			case 2:printf("Enter the position where you want to insert new node: ");
			scanf("%d", &n);
```

```
                        printf("Enter data of %d node : ", n);
                        scanf("%d", &data);
                                insert_at_anyposition(data,n);
                                break;
                        case 3:search();
                                break;
                        case 4:delete_at_end();
                                break;
                        case 5:display();
                                break;
                        case 6:exit(0);
                                break;
                        default: printf("Enter a valid choice");
                }
        }
}

void createList(int n)
{
   int i, data;
   struct node *newNode;

   if(n >= 1)
   {
```

```
head = (struct node *)malloc(sizeof(struct node));

printf("Enter data of 1 node: ");
scanf("%d", &data);

head->data = data;
head->prev = NULL;
head->next = NULL;

last = head;
for(i=2; i<=n; i++)
{
   newNode = (struct node *)malloc(sizeof(struct node));

   printf("Enter data of %d node: ", i);
   scanf("%d", &data);

   newNode->data = data;
   newNode->prev = last;
   newNode->next = NULL;

   last->next = newNode;
   last = newNode;
}
```

```
        printf("\nDOUBLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
void insert_at_anyposition(int data,int position)
{
        int i;
    struct node * newNode, *temp;

    if(head == NULL)
    {
        printf("Error, List is empty!\n");
    }
    else
    {
        temp = head;
        i=1;

        while(i<position-1 && temp!=NULL)
        {
            temp = temp->next;
            i++;
        }
```

```
      if(temp!=NULL)
      {
         newNode = (struct node *)malloc(sizeof(struct node));

         newNode->data = data;
         newNode->next = temp->next;
         newNode->prev = temp;
         if(temp->next != NULL)
         {
           temp->next->prev = newNode;
         }

         temp->next = newNode;

         printf("NODE INSERTED SUCCESSFULLY AT %d POSITION\n", position);
      }
      else
      {
         printf("Error, Invalid position\n");
      }
   }
}

void search()
```

```
{
        struct node *ptr;
        int item,i=0,flag;
        ptr = head;
        if(ptr == NULL)
        {
                printf("\nEmpty list\n");
        }
        else
        {
                printf("\nEnter the value of node you want to search:\n");
                scanf("%d",&item);
                while (ptr!=NULL)
                {
                        if(ptr->data == item)
                        {
                                printf("\nNode found at %d position\n ",i+1);
                                flag=0;
                                break;
                        }
                        else
                        {
                                flag=1;
                        }
```

```
                i++;
                ptr = ptr -> next;
                }
                if(flag==1)
                {
                        printf("\nNode not found\n");
                }
        }
}

void delete_at_end()
{
        struct node *ptr;
        if(head == NULL)
        {
                printf("\nCannot delete");
        }
        else if(head->next == NULL)
        {
                head = NULL;
                free(head);
                printf("\nNode deleted\n");
        }
        else
```

```
        {
                ptr = head;
                if(ptr->next != NULL)
                {
                ptr = ptr -> next;
                }
                ptr -> prev -> next = NULL;
                free(ptr);
                printf("\nNode deleted\n");
        }
 }

void display()
{
        struct node *ptr;
        if(head == NULL)
        {
        printf("List is empty");
        }
        else
        {
        printf("\n The nodes in DoublyLL : \n");
        ptr = head;
        while(ptr != NULL)
```

```
        {
                printf("%d\n",ptr->data);
                ptr=ptr->next;
        }
        }
}
```

a)node insert at particular position

```
1.Create the list
2.Insert at particular position
3.Search an element
4.Delete an element at end
5.Display
6.Exit
Enter your choice : 2
Enter the position where you want to insert new node: 3
Enter data of 3 node : 12
NODE INSERTED  AT 3 POSITION

1.Create the list
2.Insert at particular position
3.Search an element
4.Delete an element at end
5.Display
6.Exit
Enter your choice : 5
```

b)search an element

```
 The nodes in DoublyLL :
8
3
12
2

1.Create the list
2.Insert at particular position
3.Search an element
4.Delete an element at end
5.Display
6.Exit
Enter your choice : 3

Enter the value of node you want to search:
8

Node found at 1 position
```

c)delete at end

```

1.Create the list
2.Insert at particular position
3.Search an element
4.Delete an element at end
5.Display
6.Exit
Enter your choice : 4

Node deleted
```