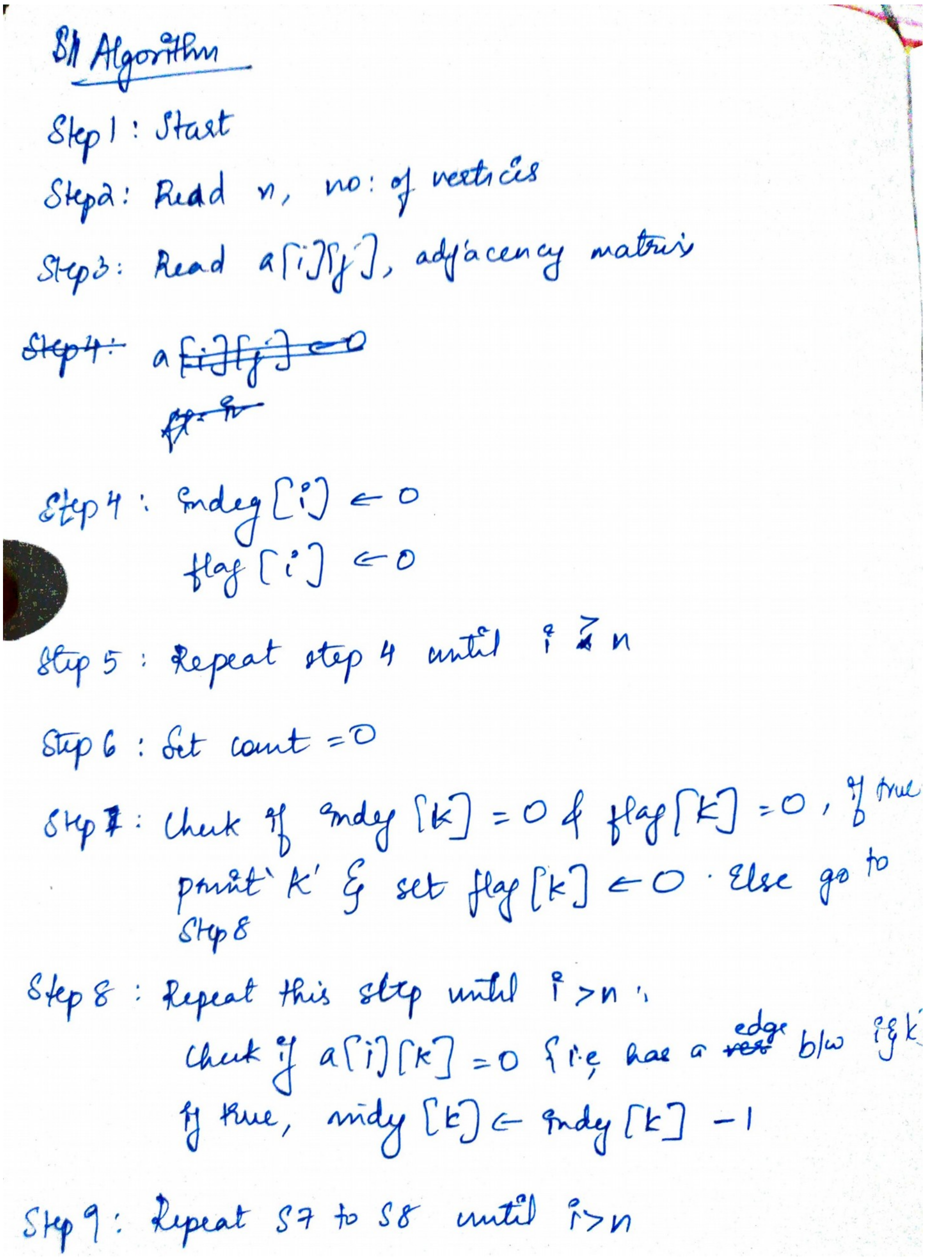
S1 EXTERNAL LAB EXAMINATION
ADVANCED DATA STRUCTURES

ATTENDED BY:
NIDHI RAJ
TKM20MCA2025

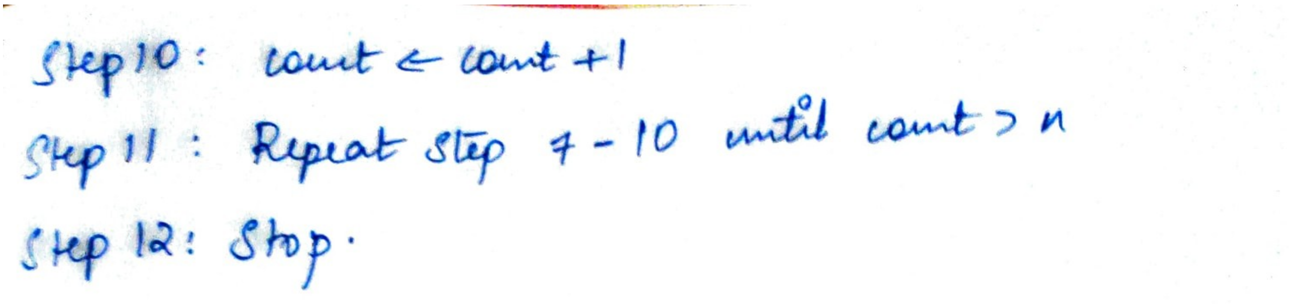
QUESTION 1:

Consider a directed acyclic graph G given in the following figure.Develop a program to implement topological sort.

ALGORITHM:



```
S1 Algorithm

Step1 : Start
Step2: Read n, no: of vertices
Step3: Read a[i][j], adjacency matrix
~~Step4:~~ a ~~[i][j] ← 0~~
   ~~ff ← n~~

Step 4 : indeg[i] ← 0
         flag[i] ← 0

Step 5 : Repeat step 4 until i ≥ n

Step 6 : Set count = 0

Step 7 : Check if indeg[k] = 0 & flag[k] = 0, if true
         print 'k' & set flag[k] ← 0 . Else go to
         Step8

Step 8 : Repeat this step until i > n :
         Check if a[i][k] = 0 {i.e has a ~~vert~~ edge b/w i & k
         If true, indeg[k] ← indeg[k] -1

Step 9 : Repeat S7 to S8 until i>n
```



Step 10 : count ← count +1

Step 11 : Repeat step 7 - 10 until count > n

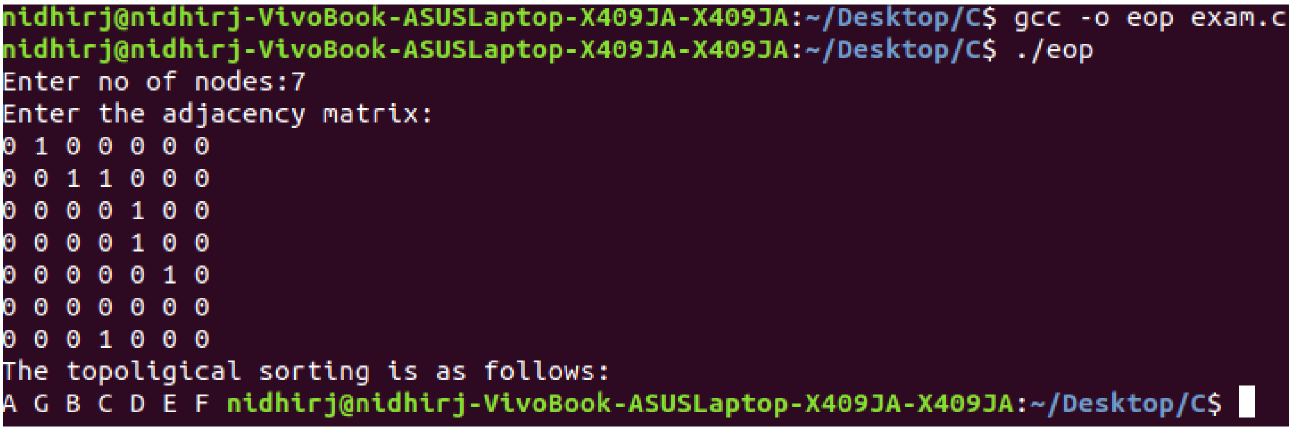
Step 12: Stop.

PROGRAM CODE:

```
#include<stdio.h>
void main()
{
   int n,a[10][10],indeg[10],flag[10],i,j,k,count=0;
   printf("Enter no of nodes:");
   scanf("%d",&n);
   printf("Enter the adjacency matrix:\n");
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         scanf("%d",&a[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]+a[j][i];
      }
   }
   printf("The topoligical sorting is as follows:\n");
   while(count<n)
   {
      for(k=0;k<n;k++)
      {
         if(indeg[k]==0 && flag[k]==0)
         {
            printf("%c ",(k+65));
            flag[k]=1;
         }
         for(i=0;i<n;i++)
         {
             if(a[i][k])
```

```
                {
                    indeg[k]--;
                }
            }
        }
        count++;
    }
}
```

OUTPUT



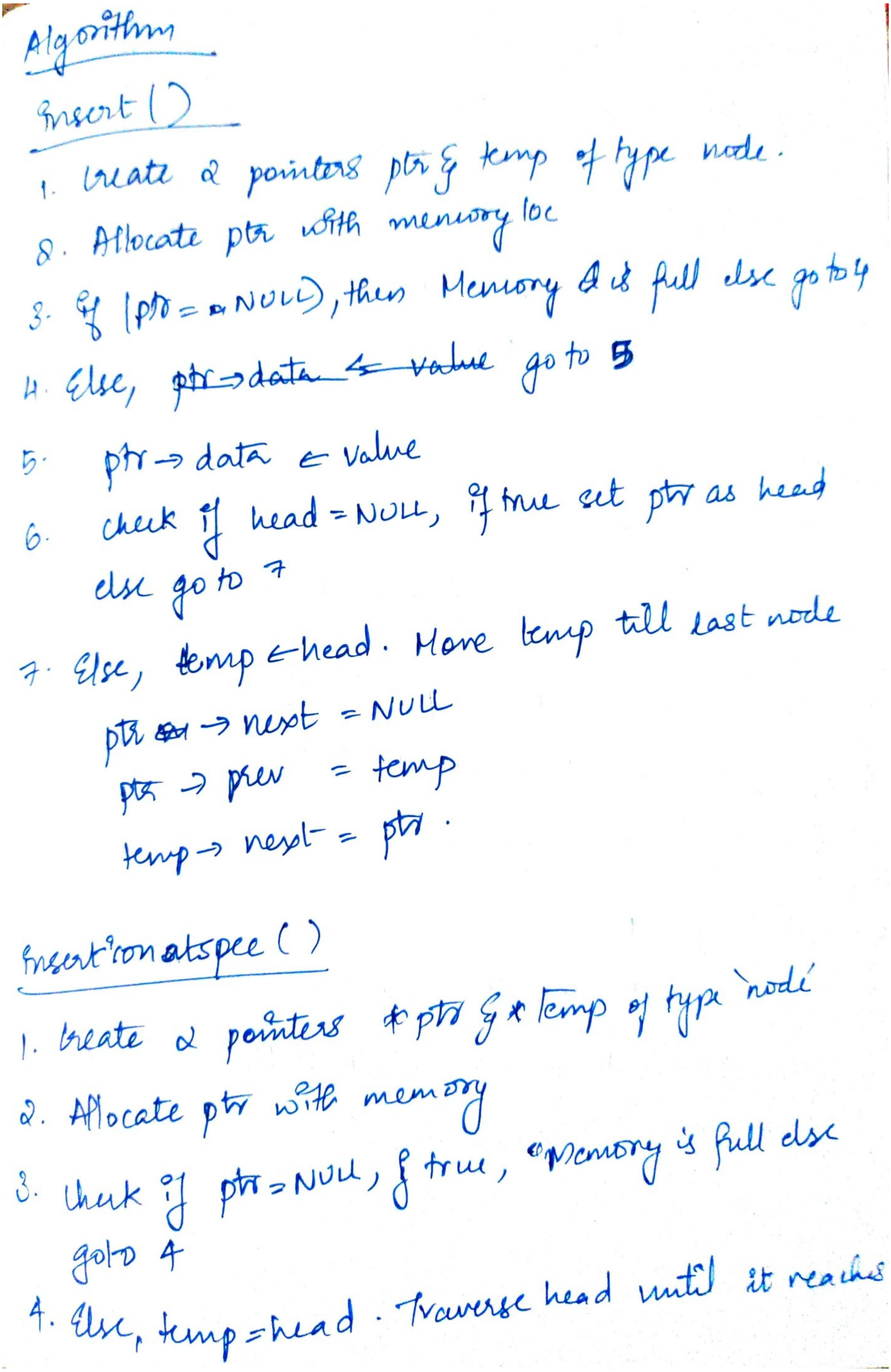
```
nidhirj@nidhirj-VivoBook-ASUSLaptop-X409JA-X409JA:~/Desktop/C$ gcc -o eop exam.c
nidhirj@nidhirj-VivoBook-ASUSLaptop-X409JA-X409JA:~/Desktop/C$ ./eop
Enter no of nodes:7
Enter the adjacency matrix:
0 1 0 0 0 0 0
0 0 1 1 0 0 0
0 0 0 0 1 0 0
0 0 0 0 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
The topoligical sorting is as follows:
A G B C D E F nidhirj@nidhirj-VivoBook-ASUSLaptop-X409JA-X409JA:~/Desktop/C$
```

QUESTION 2:

WAP for creating a doubly linked list and perform the following:

1. Insert at element at particular position
2. Serach an element
3. Delete an element at end

ALGORITHM:



Algorithm

Insert ( )

1. Create 2 pointers ptr & temp of type node.
2. Allocate ptr with memory loc
3. If (ptr = NULL), then Memory ~~A~~ is full else goto 4
4. Else, ~~ptr→data ← value~~ go to 5
5. ptr → data ← value
6. check if head = NULL, if true set ptr as head else go to 7
7. Else, temp ← head. Move temp till last node

    ptr → next = NULL

    ptr → prev = temp

    temp → next = ptr.

Insertionatspec ( )

1. Create 2 pointers *ptr & *temp of type 'node'
2. Allocate ptr with memory
3. Check if ptr = NULL, if true, Memory is full else goto 4
4. Else, temp = head. Traverse head until it reaches

specified pos. Then set

ptr -> data = value

ptr -> next = temp -> next

ptr -> prev = temp

temp -> next -> prev = ptr

temp -> next = ptr.

Search()

```
1. i ← 0, flag ← 0
2. ~~if~~ check if head = NULL, if true, DLL is empty else g
   3.
3. Create a pointer temp; temp ← head.
4. Check if temp -> data = value, if true print the locat
   & make flag = 0. Else goto 5
5. Else, flag ← 1
6. i ← i+1. ~~ptr~~ temp ~~->~~ = temp -> next.
7. After checking throughout, check if flag =1, then
   print "Element is not found".
```

delete~~node~~atend ()

1. Check if head = NULL, if true print "DLL is empty". Else goto 2

2. Create a pointer temp & temp ← head

3. traverse temp until it reaches last node.

4. item ← temp→data

5. Set temp → prev → next = NULL (setting it as last node)

6. Remove temp.

Display

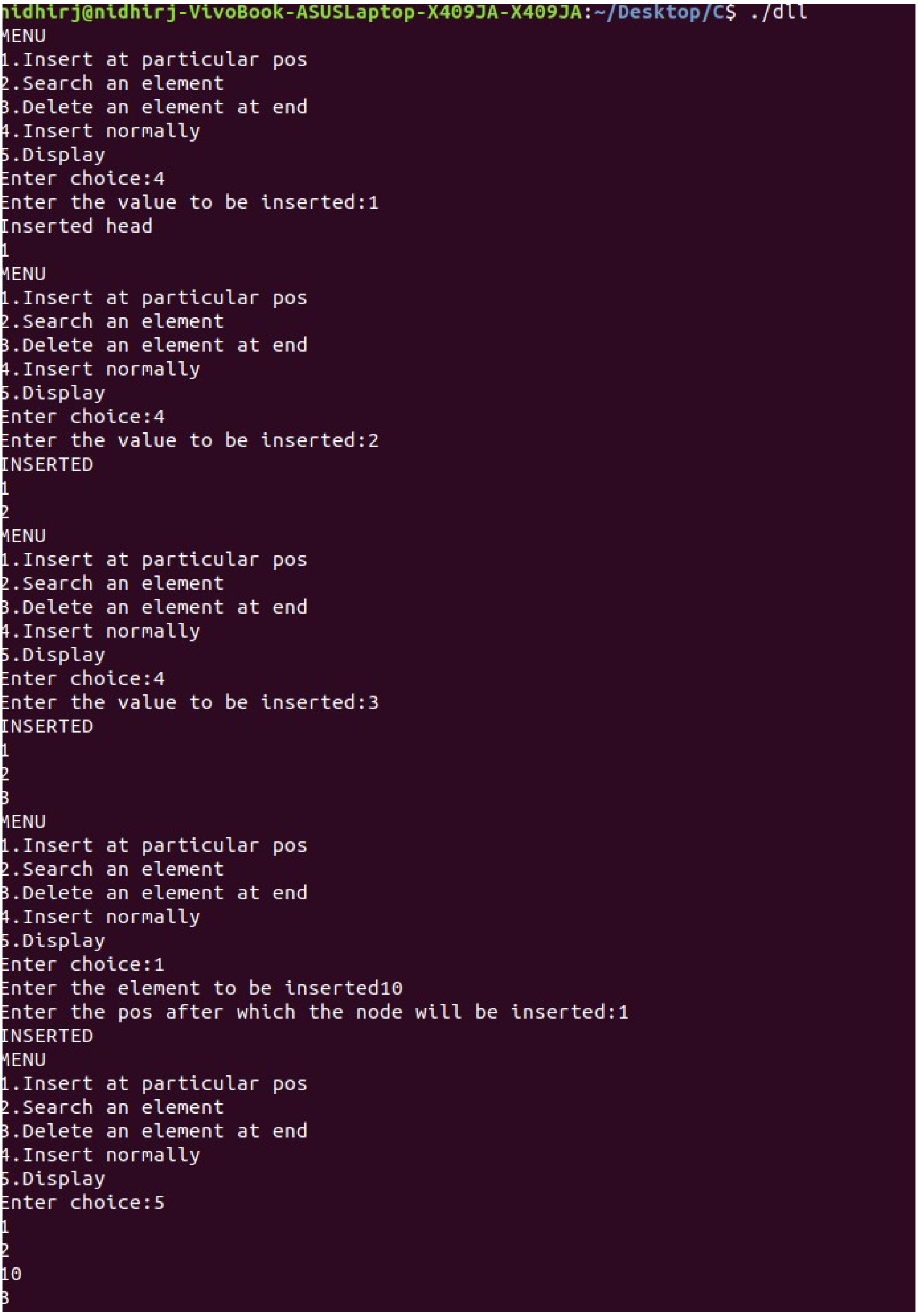
1. Create a pointer ptr (type = node) & ptr ← head.

2. traverse until ptr reaches end node & in between print, ptr→data (Data at each node).

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void insert(int value)
{
   struct node *ptr,*temp;
   ptr = (struct node *) malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
       ptr->data=value;
      if(head == NULL)
      {
          ptr->next = NULL;
          ptr->prev = NULL;
          head = ptr;
          printf("Inserted head\n");
      }
      else
      {
         temp = head;
         while(temp->next!=NULL)
         {
             temp = temp->next;
         }
         temp->next = ptr;
         ptr ->prev=temp;
         ptr->next = NULL;
         printf("INSERTED\n");
      }
   }
}
void insertionatspec(int value,int pos)
{
    struct node *ptr,*temp;
    int i;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\n OVERFLOW");
    }
    else
    {
        temp=head;
        for(i=0;i<pos;i++)
        {
            temp = temp->next;
        }
        ptr->data = value;
        ptr->next = temp->next;
        ptr -> prev = temp;
        temp->next = ptr;
        temp->next->prev=ptr;
        printf("INSERTED\n");
    }
}
void search(int value)
{
    int i=0,flag=0;
    if(head==NULL)
    {
        printf("DLL IS EMPTY\n");
    }
    else
    {
        struct node *temp=head;
        if(temp->data==value)
        {
            printf("Data found at location %d\n",i+1);
            flag=0;
            exit(0);
        }
        else
        {
            flag=1;
        }
        i++;
        temp=temp->next;
    }
    if(flag==1)
    {
        printf("%d is not found\n",value);
    }
}
void deleteatend()
{
    if(head==NULL)
    {
        printf("DLL is empty\n");
    }
    else
    {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        int item=temp->data;
        temp->prev->next=NULL;
        free(temp);
        printf("\n%d is deleted\n",item);
    }
}
void display()
{
    struct node *ptr;
    ptr = head;
    while(ptr != NULL)
    {
        printf("%d\n",ptr->data);
        ptr=ptr->next;
    }
}
void main()
{
    int ch,value,pos,s;
    do
    {
        printf("MENU\n");
        printf("1.Insert at particular pos\n2.Search an element\n3.Delete an element at 
end\n4.Insert normally\n5.Display\n");
        printf("Enter choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
            {
                printf("Enter the element to be inserted");
                scanf("%d",&value);
                printf("Enter the pos after which the node will be inserted:");
                scanf("%d",&pos);
                insertionatspec(value,pos);
                break;
            }
            case 2:
            {
                printf("Enter the value to be searched:");
                scanf("%d",&s);
                search(s);
                break;
            }
            case 3:
            {
                deleteatend();
                break;
            }
            case 4:
            {
                printf("Enter the value to be inserted:");
                scanf("%d",&value);
                insert(value);
            }
            case 5:
            {
                display();
                break;
            }
            default:
            {
                printf("INVALID INPUT");
                break;
            }
        }
    }while(ch!=0 && ch<6);

}
```

OUTPUT:



```
nidhirj@nidhirj-VivoBook-ASUSLaptop-X409JA-X409JA:~/Desktop/C$ ./dll
MENU
1.Insert at particular pos
2.Search an element
3.Delete an element at end
4.Insert normally
5.Display
Enter choice:4
Enter the value to be inserted:1
Inserted head
1
MENU
1.Insert at particular pos
2.Search an element
3.Delete an element at end
4.Insert normally
5.Display
Enter choice:4
Enter the value to be inserted:2
INSERTED
1
2
MENU
1.Insert at particular pos
2.Search an element
3.Delete an element at end
4.Insert normally
5.Display
Enter choice:4
Enter the value to be inserted:3
INSERTED
1
2
3
MENU
1.Insert at particular pos
2.Search an element
3.Delete an element at end
4.Insert normally
5.Display
Enter choice:1
Enter the element to be inserted10
Enter the pos after which the node will be inserted:1
INSERTED
MENU
1.Insert at particular pos
2.Search an element
3.Delete an element at end
4.Insert normally
5.Display
Enter choice:5
1
2
10
3
```

GITHUB LINK:https://github.com/NidhiRj/EXAMDS