DATA STRUCTURE LAB

ANANATHAPADMANABHAN S

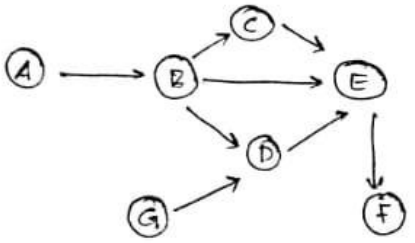
S1 MCA

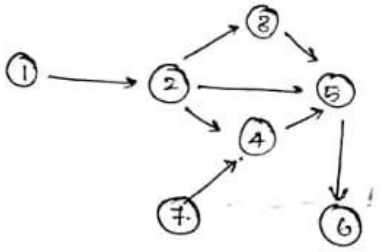
TKM20MCA-2007

Git Hub : https://github.com/ananthapadmanabhans/Data-Structure/tree/main/S1%20Lab%20Exam

1.Consider a directed acyclic graph G.

Perform Topological sorting.





Adj-Matrix Representation

No. of vertices — 7

	1	2	3	4	5	6	7.
1	0	1	0	0	0	0	0
2	0	0	1	1	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0

Algorithm

1. Start
2. Initialize adj matrix to a [20] [20]
3. Make all value of arrays id and fl as 0
4. for (i=0; i<7; i++)
   { for (j=0; j<7; j++)
   { id [i] = id[i] + a[j][i]; }}
5. while (c<n)
   part { for (k=0; k<7; k++)
   { if ((id [k]==0) && (fl[k]==0))
   { part k+1
   fl[k]=1;
   }
   if (a[i][k]==1)
   id[k]--;
   c++;
6. Stop.

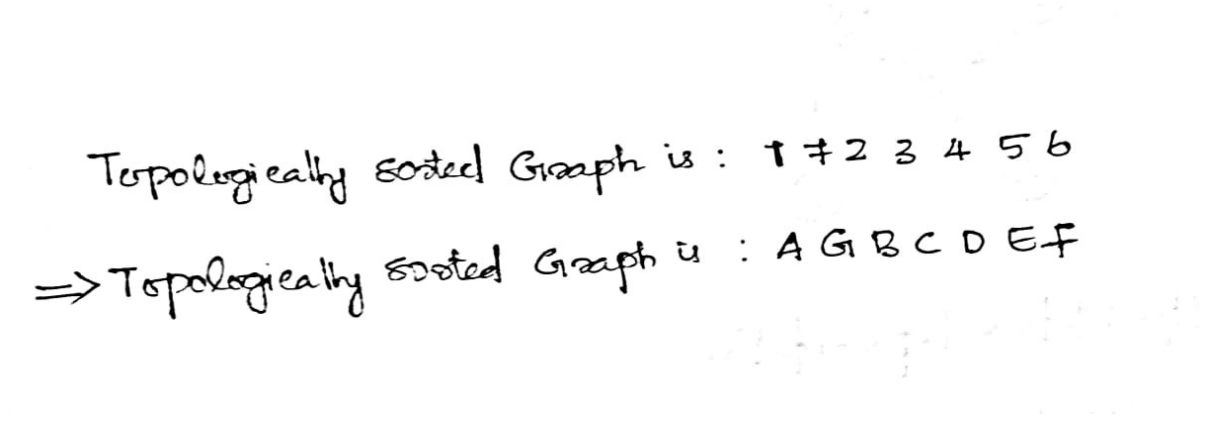
Program code:

```
#include <stdio.h>

int main(){
int i,j,k,id[10],fl[10],c=0;
int a[10][10]={
   {0,1,0,0,0,0,0},
   {0,0,1,1,1,0,0},
   {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}};

for(i=0;i<7;i++)
{
id[i]=0;
 fl[i]=0;
}

for(i=0;i<7;i++)
for(j=0;j<7;j++)
id[i]=id[i]+a[j][i];

printf("\nTopologically Sorted Graph is :");

while(c<7){
for(k=0;k<7;k++){
if((id[k]==0) && (fl[k]==0)){
printf("%d ",(k+1));
fl[k]=1;
```

```
}

for(i=0;i<7;i++){
if(a[i][k]==1)
id[k]--;
}
}

 c++;
}

return 0;
}
```

Output:

```
ananthapadmanabhan@ananthapadmanabhan-VirtualBox:~/Desktop$ gcc -o dlab dlab.c
ananthapadmanabhan@ananthapadmanabhan-VirtualBox:~/Desktop$ ./dlab

Topologically Sorted Graph is :1 7 2 3 4 5 6 ananthapadmanabhan@ananthapadmanabhan-VirtualBox:~/Desktop$
```



2.Create a doubly linked list and perform operations:

a. Insert

b. search

c. delete

Algorithm

1. Start.
2. print Choose an option
3. print
    1. insert at beginning
    2. insert at last
    3. insert at any location
    4. delete from beginning
    5. delete from last
    6. Search
    7. Display
    8. Exit.

4. Case 1:
    Call insertion_beginning()

Case 2:
    Call insertion_last()

Case 3:
    Call insertion specified()

Case 4:
    call. deletion_begin

Case 5:
    Call deletion_last()

Case 6:
    Call search()

Case 7:
    Call display()

Case 8
    exit(0)

Insertion : Begin

1. if start = null
   start = t.
2. else
   t → next = null
   t → next → prep = t
   start = t.
   return

Middle

3. print "enter location"
2 read x
3 p = start
4 repeat while p-null
5 if p → info == n
   t → next = p → next
   p → next = t.
   t => prev = p
   p => next → prev = t.
   return
6 else
   p = p → next.
   print "not found"
   t → next = null
   p → next = t

Deletion :
Begining

1. p = start.
2 p → next → prev = null
3 start = p → next
4. start = p → next
   delnode (p)

Middle

1. read node to be deleted x
2. p = start
3. repeat until p <> null
   if (p → info = x).
      p → prev → next ⇒ p → next
      p → next → prev = p → prev
      del node (p).
   else
      p = p → next.
5. print not found.

Last

1. p - start
2. repeat while p <> null
      if p → next = null
         del node (p)
3. return.

Program code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 struct node *prev;
 struct node *next;
 int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();
void main ()
{
int choice =0;
 while(choice != 9)
 {

 printf("\nChoose option \n");
```

```
 printf("\n1.Insert at begining\n2.Insert at last\n3.Insert at any
location\n4.Delete from Beginning\n 5.Delete from
last\n6.Search\n7.Display\n8.Exit\n");
 printf("\nEnter your choice: ");
 scanf("\n%d",&choice);
 switch(choice)
 {
 case 1:
 insertion_beginning();
 break;
 case 2:
 insertion_last();
 break;
 case 3:
 insertion_specified();
 break;
 case 4:
 deletion_beginning();
 break;
 case 5:
 deletion_last();
 break;
 case 6:
search();
 break;
 case 7:
```

```
 display();
 break;
 case 8:
 exit(0);
 break;

 default:
 printf("Invalid !!");
 } } }
void insertion_beginning()
{
 struct node *ptr;
 int item;
 ptr = (struct node *)malloc(sizeof(struct node));
 if(ptr == NULL)
 {
 printf("\nOVERFLOW");
 }
 else
 {
 printf("\nEnter Item value = ");
 scanf("%d",&item);

 if(head==NULL)
 {
 ptr->next = NULL;
```

```
 ptr->prev=NULL;
 ptr->data=item;
 head=ptr;
 }
 else
 {
 ptr->data=item;
 ptr->prev=NULL;
 ptr->next = head;
 head->prev=ptr;
 head=ptr;
 }
 printf("\nNode inserted\n");
} }
void insertion_last()
{
 struct node *ptr,*temp;
 int item;
 ptr = (struct node *) malloc(sizeof(struct node));
 if(ptr == NULL)
 {
 printf("\nOVERFLOW");
 }
 else
 {
 printf("\nEnter the value = ");
```

```
 scanf("%d",&item);
 ptr->data=item;
 if(head == NULL)
 {
 ptr->next = NULL;
 ptr->prev = NULL;
 head = ptr;
 }
 else
 {
 temp = head;
 while(temp->next!=NULL)
 {
 temp = temp->next;
 }
 temp->next = ptr;
 ptr ->prev=temp;
 ptr->next = NULL;
 } }
 printf("\nNode inserted\n");
 }
void insertion_specified()
{
 struct node *ptr,*temp;
 int item,loc,i;
 ptr = (struct node *)malloc(sizeof(struct node));
```

```
if(ptr == NULL)
{
printf("\n OVERFLOW");
}
else
{
temp=head;
printf("Enter the location = ");
scanf("%d",&loc);
for(i=0;i<loc;i++)
{
temp = temp->next;
if(temp == NULL)
{
printf("\n There are less than %d elements", loc);
return;
} }
printf("Enter value = ");
scanf("%d",&item);
ptr->data = item;
ptr->next = temp->next;
ptr -> prev = temp;
temp->next = ptr;
temp->next->prev=ptr;
printf("\nnode inserted\n");
} }
```

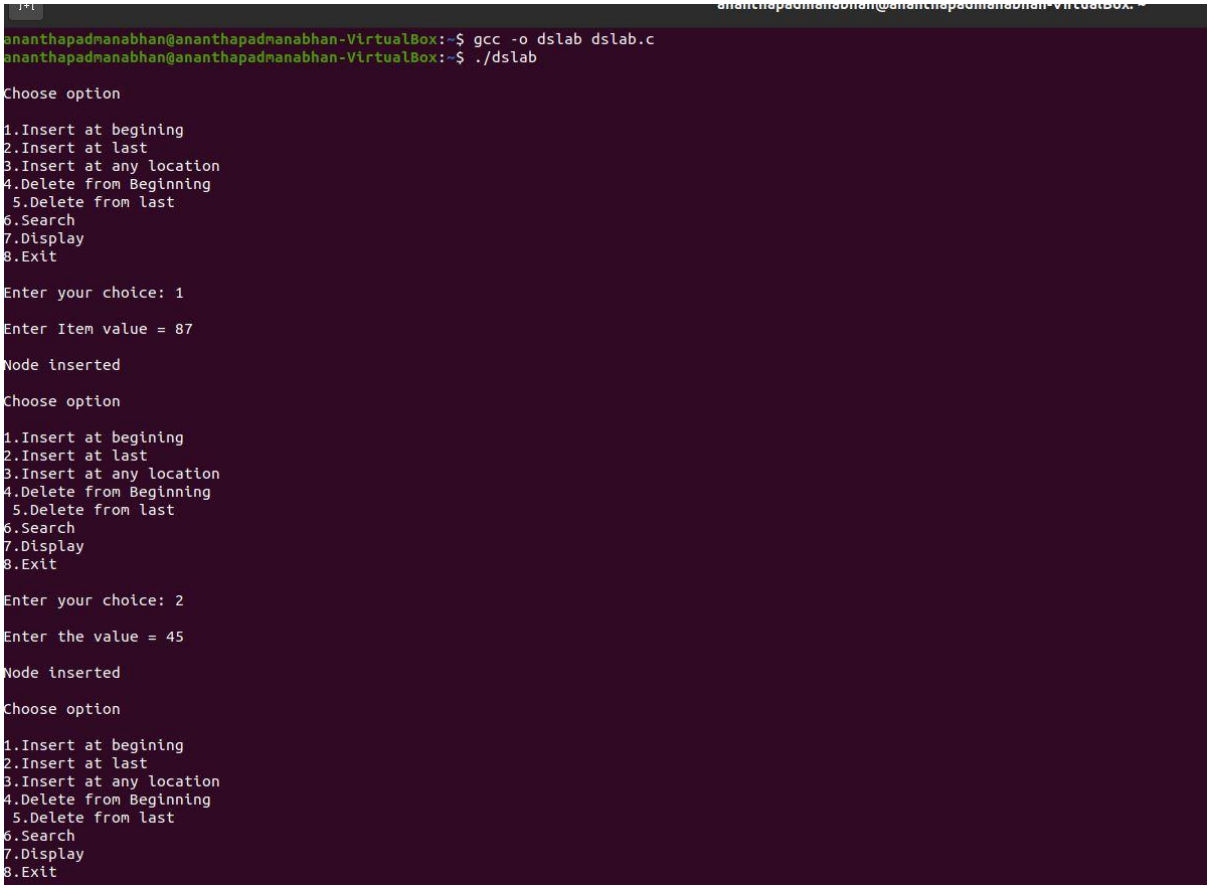
```
void deletion_beginning()
{
 struct node *ptr;
 if(head == NULL)
 {
 printf("\n UNDERFLOW");
 }
 else if(head->next == NULL)
 {
 head = NULL;
 free(head);
 printf("\nnode deleted\n");
 }
 else
 {
 ptr = head;
 head = head -> next;
 head -> prev = NULL;
 free(ptr);
 printf("\nnode deleted\n");
 } }
void deletion_last()
{
 struct node *ptr;
 if(head == NULL)
 {
```

```
 printf("\n UNDERFLOW");
 }
 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;
 printf("\n Elements are:\n");
 ptr = head;
 while(ptr != NULL)
```

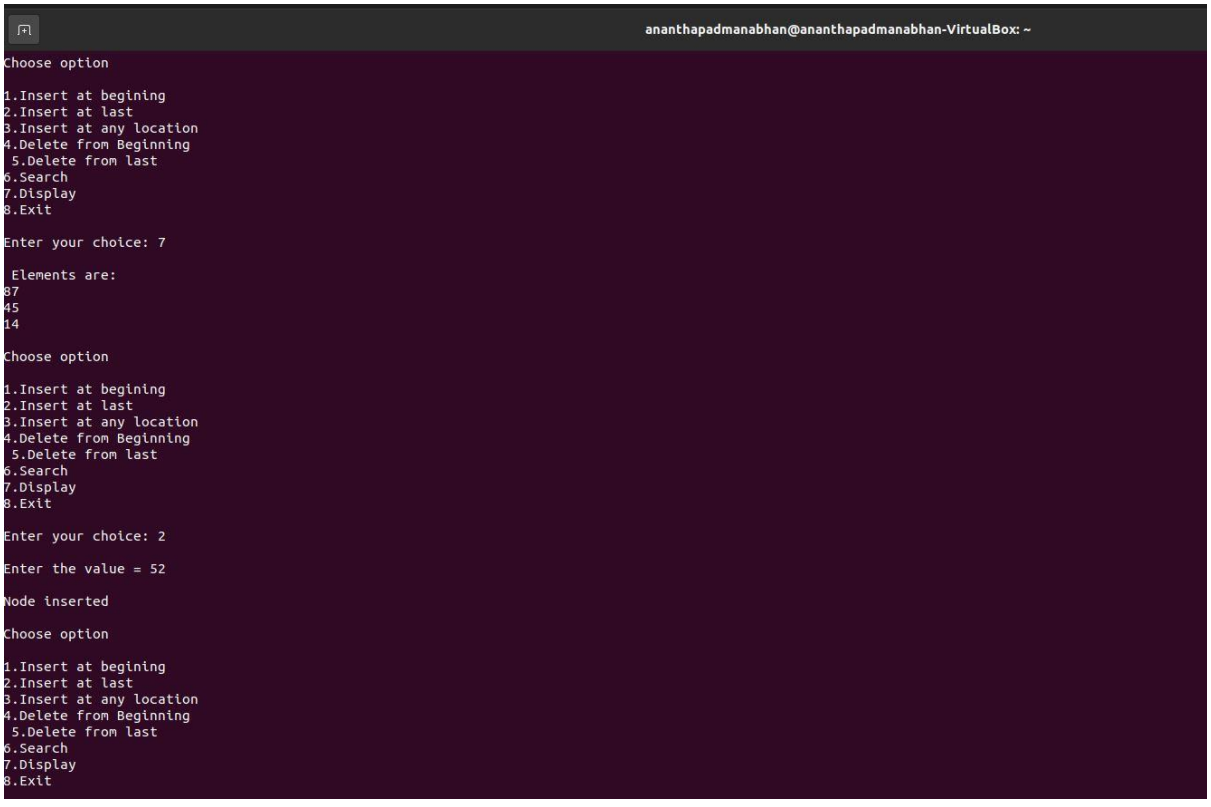
```
 {
 printf("%d\n",ptr->data);
 ptr=ptr->next;
 } }
void search()
{
 struct node *ptr;
 int item,i=0,flag;
 ptr = head;
 if(ptr == NULL)
  {
 printf("\nList is empty\n");
 }
 else
 {
 printf("\nEnter item to search?\n");
 scanf("%d",&item);
 while (ptr!=NULL)
 {
 if(ptr->data == item)
 {
 printf("\nItem found at location %d ",i+1);
 flag=0;
 break;
 }
 else
```

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

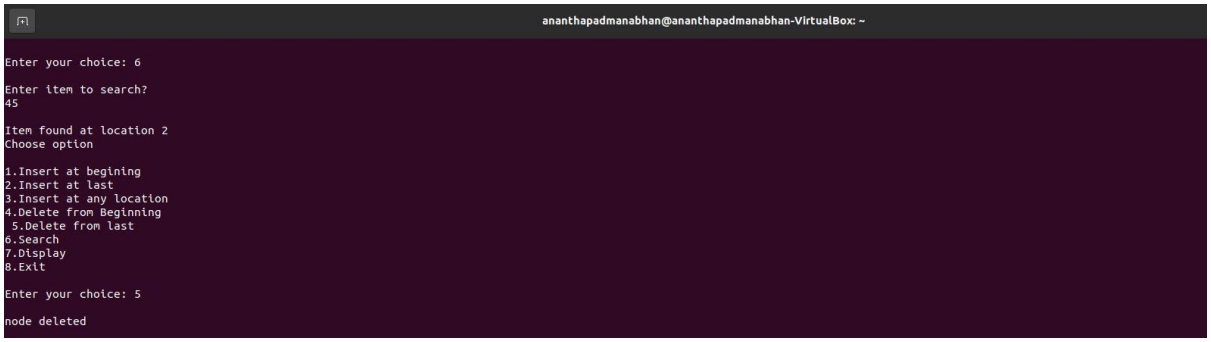
Output:



```
ananthapadmanabhan@ananthapadmanabhan-VirtualBox:~$ gcc -o dslab dslab.c
ananthapadmanabhan@ananthapadmanabhan-VirtualBox:~$ ./dslab

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit

Enter your choice: 1

Enter Item value = 87

Node inserted

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit

Enter your choice: 2

Enter the value = 45

Node inserted

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit
```



```
ananthapadmanabhan@ananthapadmanabhan-VirtualBox: ~

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit

Enter your choice: 7

 Elements are:
87
45
14

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit

Enter your choice: 2

Enter the value = 52

Node inserted

Choose option

1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
 5.Delete from last
6.Search
7.Display
8.Exit
```


ananthapadmanabhan@ananthapadmanabhan-VirtualBox: ~
Enter your choice: 6
Enter item to search?
45
Item found at location 2
Choose option
1.Insert at begining
2.Insert at last
3.Insert at any location
4.Delete from Beginning
5.Delete from last
6.Search
7.Display
8.Exit
Enter your choice: 5
node deleted