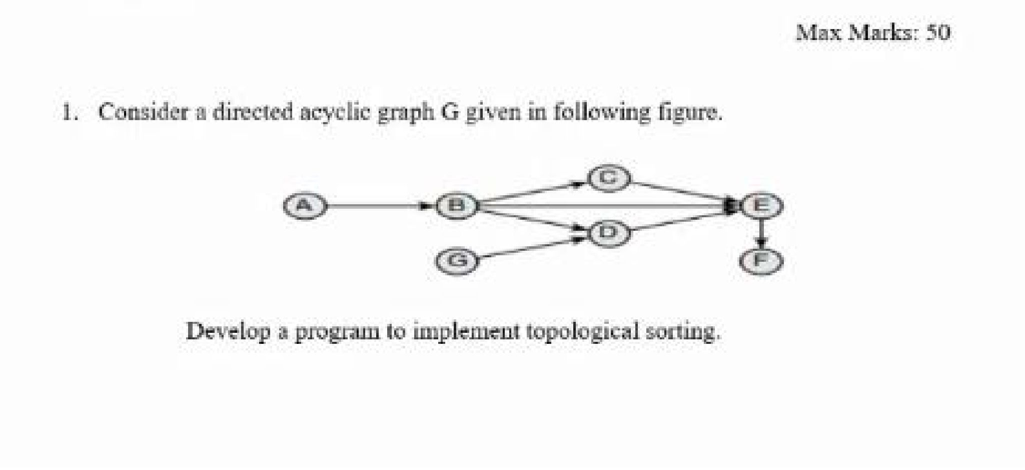
ABHILASH JOHN
TKM20MCA-2001

Data structure lab
1/7/2021

GIt Link : https://github.com/Abhilash2015mca/Data-structures/tree/main/DS%20LAB%20EXAM

Q.1:

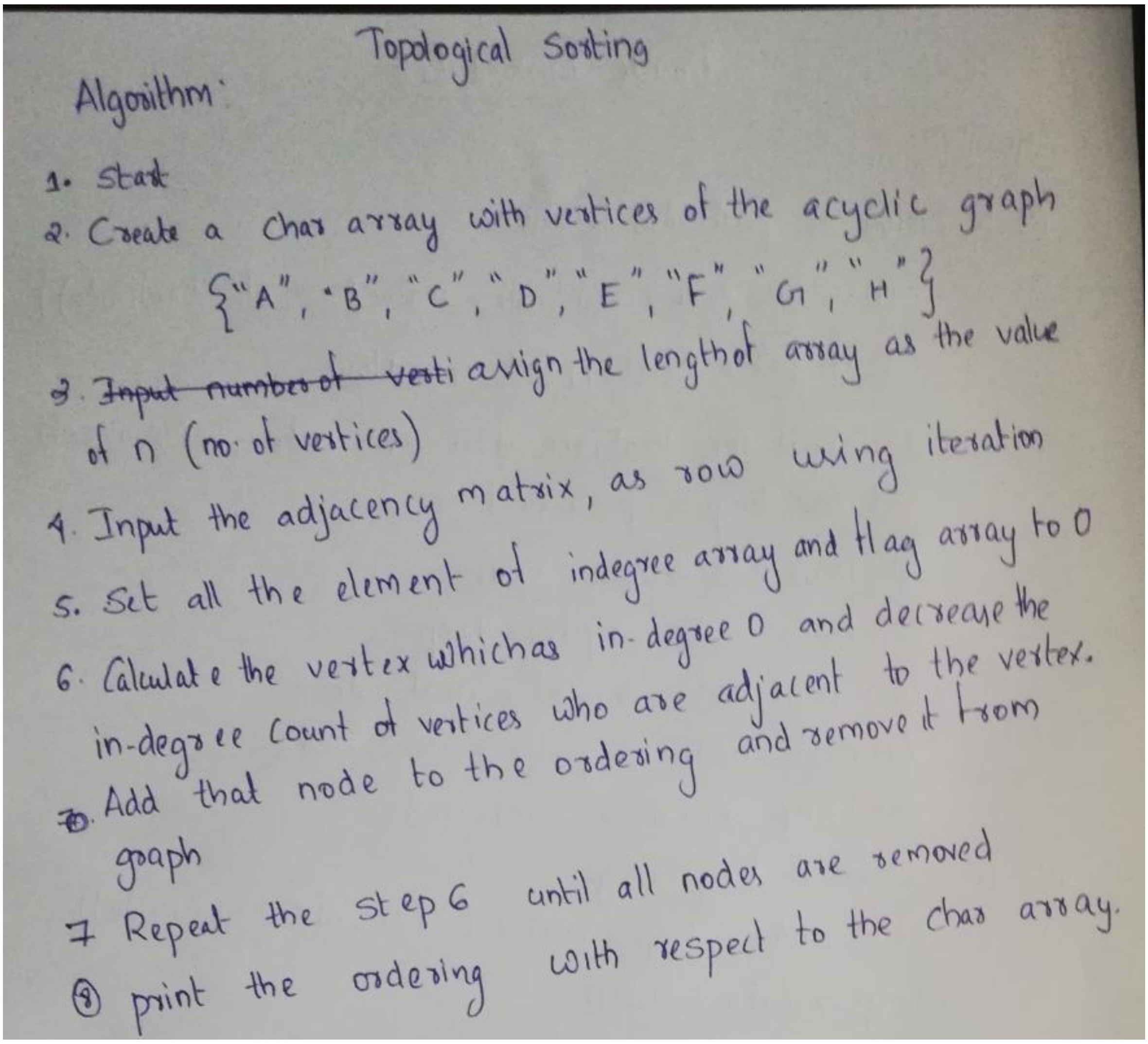


Max Marks: 50

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

Develop a program to implement topological sorting.

ALgorithm:-

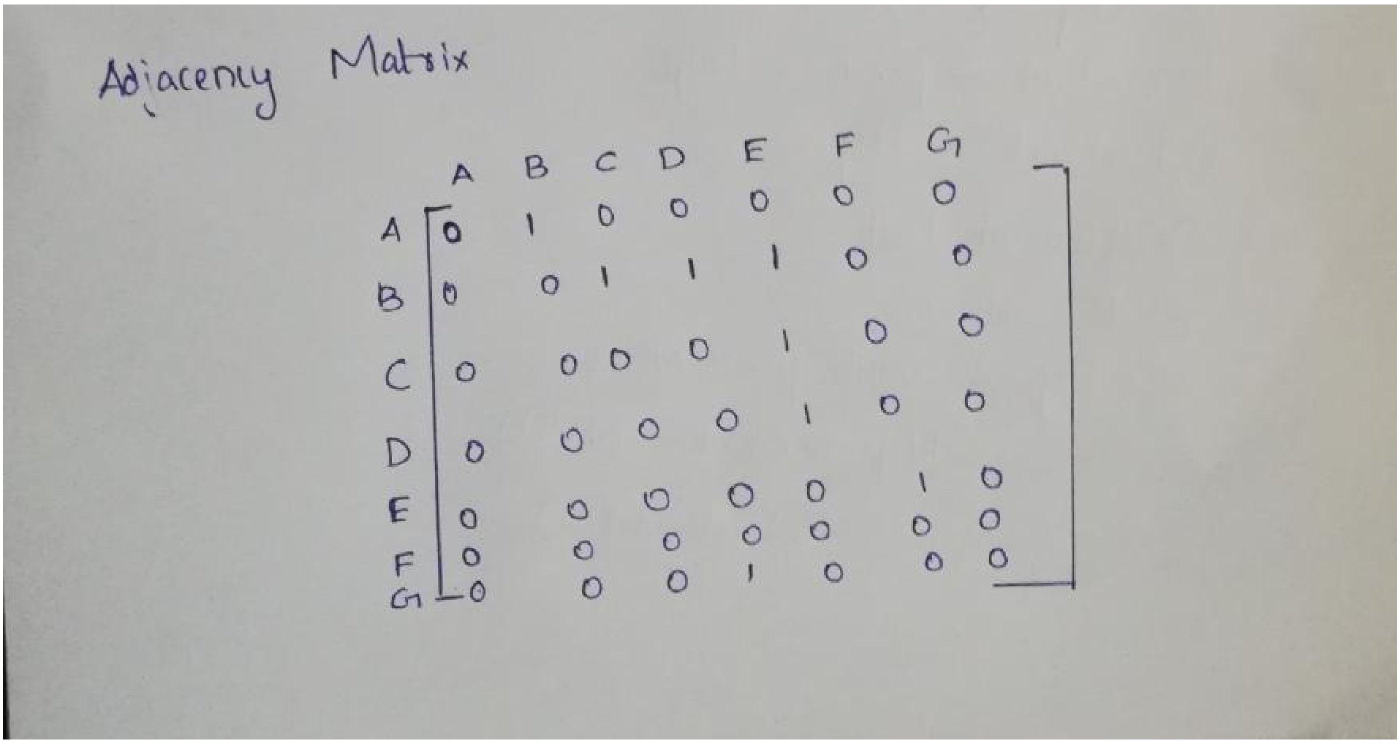


Topological Sorting

Algorithm:

1. Start
2. Create a Char array with vertices of the acyclic graph
{"A", "B", "C", "D", "E", "F", "G", "H"}
3. ~~Input number of verti~~ assign the length of array as the value
of n (no. of vertices)
4. Input the adjacency matrix, as row using iteration
5. Set all the element of indegree array and flag array to 0
6. Calculate the vertex which has in-degree 0 and decrease the
in-degree count of vertices who are adjacent to the vertex.
and remove it from
graph
7. Add that node to the ordering
7 Repeat the step 6 until all nodes are removed
8 print the ordering with respect to the char array.

Adjacency Matrix:



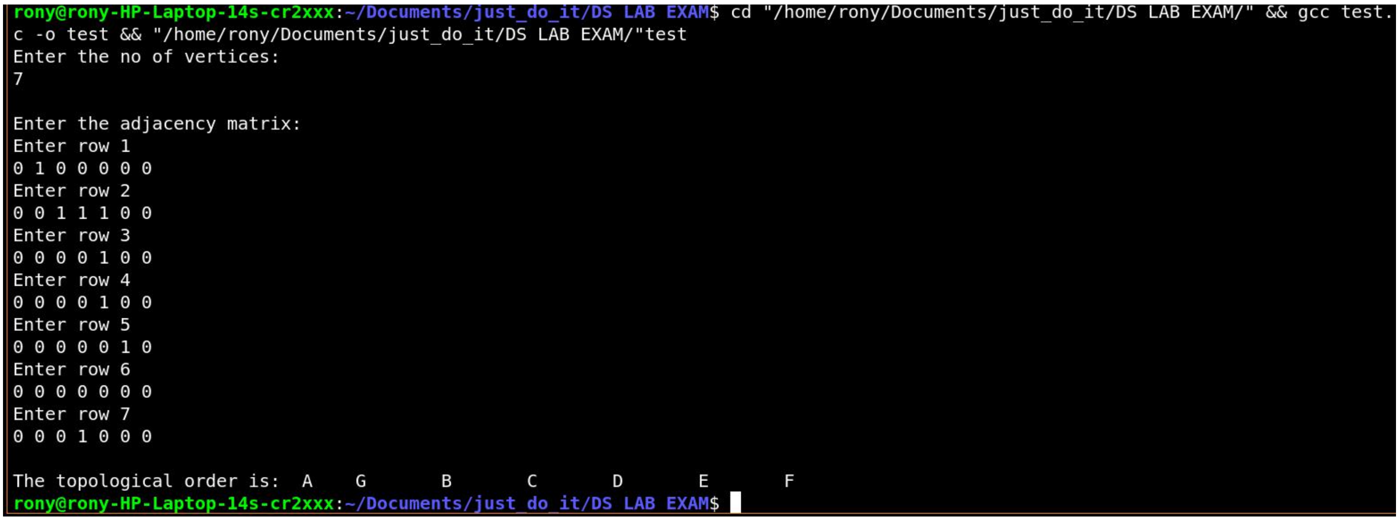
Adjacency Matrix

$$
\begin{array}{c|ccccccc}
 & A & B & C & D & E & F & G \\
\hline
A & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
C & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
D & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}
$$

Program:

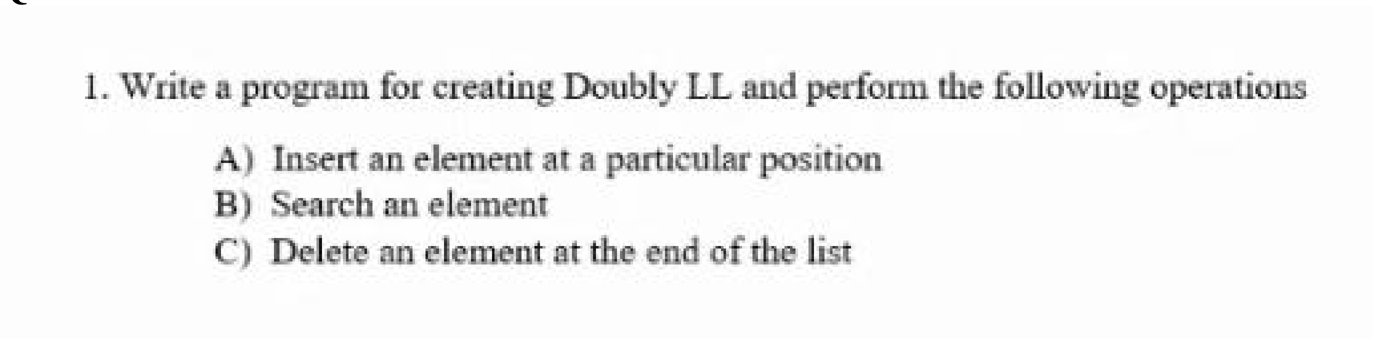
```
#include <stdio.h>

int main(){
int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;
char arr1[] = { 'A', 'B', 'C', 'D', 'E', 'F','D' };

printf("Enter the no of vertices:\n");
scanf("%d",&n);
printf("\n");

printf("Enter the adjacency matrix:\n");
for(i=0;i<n;i++){
printf("Enter row %d\n",i+1);
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("\nThe topological order is:  ");

    while(count<n){
        for(k=0;k<n;k++){
            if((indeg[k]==0) && (flag[k]==0)){
                printf("%c\t",arr1[k]);
                flag [k]=1;
            }

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

        count++;
    }

    return 0;
}
```

Result:



```
rony@rony-HP-Laptop-14s-cr2xxx:~/Documents/just_do_it/DS LAB EXAM$ cd "/home/rony/Documents/just_do_it/DS LAB EXAM/" && gcc test.
c -o test && "/home/rony/Documents/just_do_it/DS LAB EXAM/"test
Enter the no of vertices:
7

Enter the adjacency matrix:
Enter 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:  A    G       B       C       D       E       F
rony@rony-HP-Laptop-14s-cr2xxx:~/Documents/just_do_it/DS LAB EXAM$ 
```

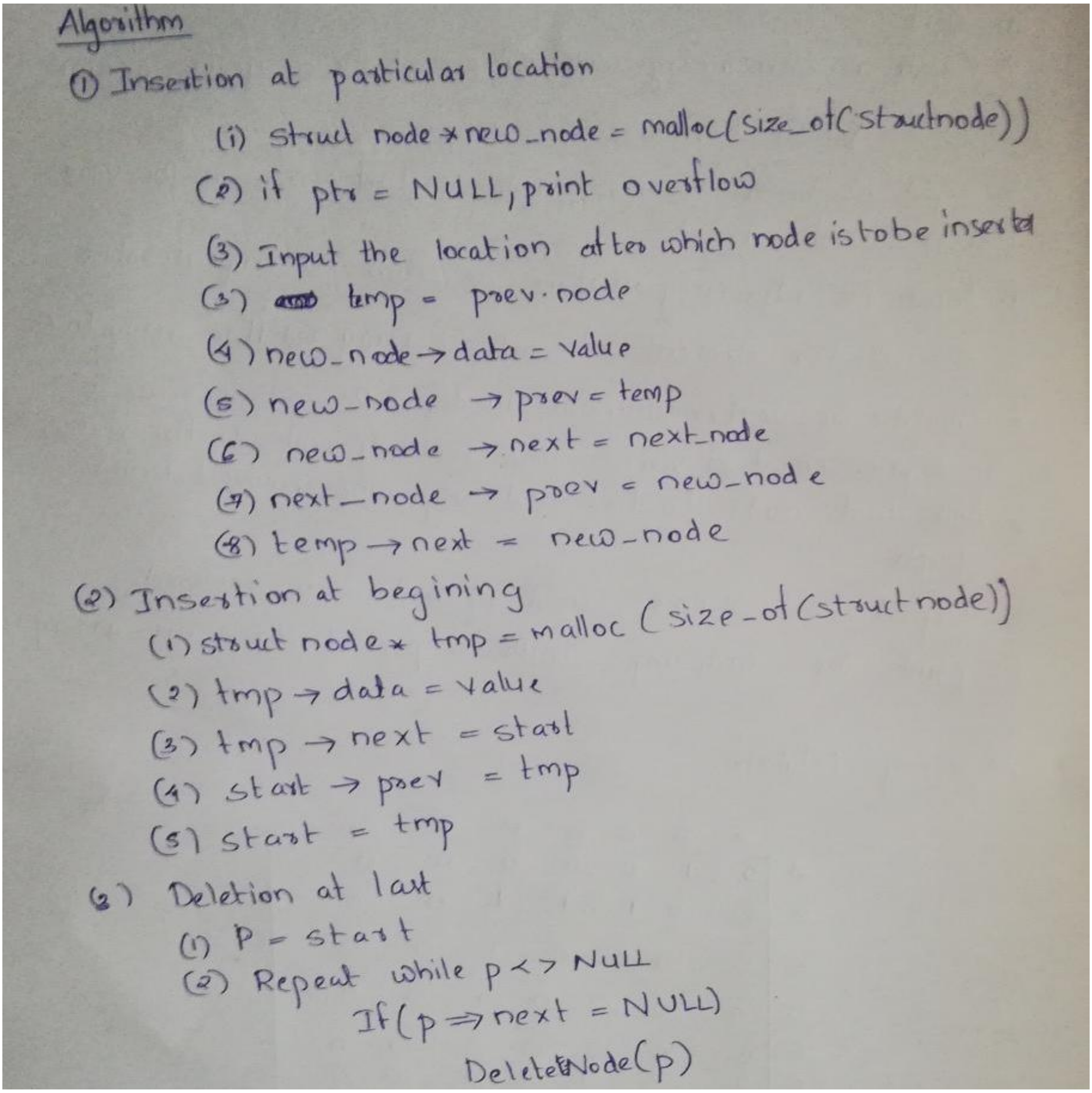
Q.2:



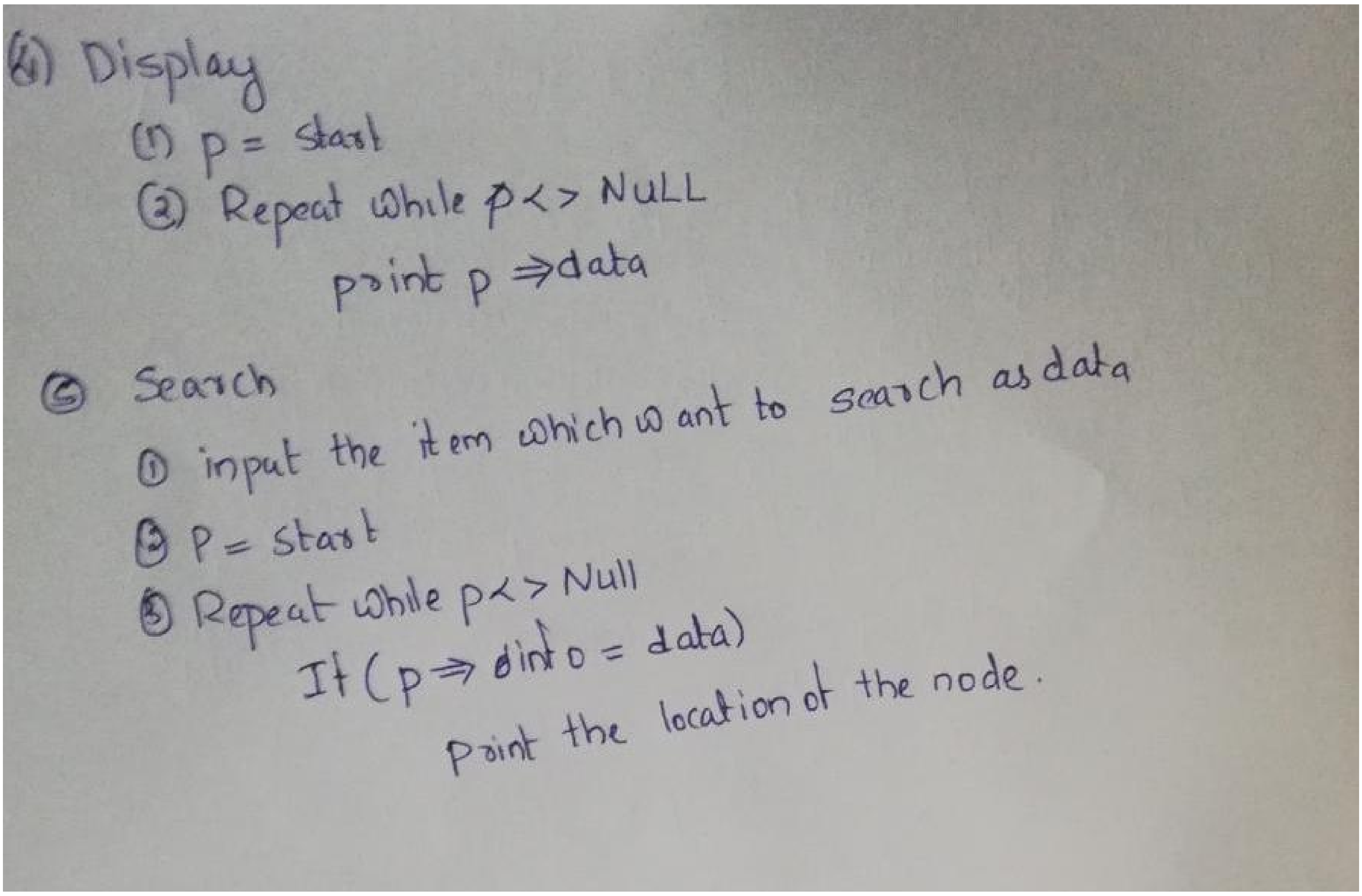
1. Write a program for creating Doubly LL and perform the following operations

   A) Insert an element at a particular position
   B) Search an element
   C) Delete an element at the end of the list

Algorithm:



```
Algorithm
① Insertion at particular location
    (i) struct node *new_node = malloc(size_of(structnode))
    (2) if ptr = NULL, print overflow
    (3) Input the location after which node is tobe inserted
    (3) temp = prev.node
    (4) new_node→data = value
    (5) new_node → prev = temp
    (6) new_node → next = next_node
    (7) next_node → prev = new_node
    (8) temp → next = new_node
(2) Insertion at begining
    (1) struct node* tmp = malloc (size-of(struct node))
    (2) tmp → data = value
    (3) tmp → next = start
    (4) start → prev = tmp
    (5) start = tmp
(3) Deletion at last
    (1) P = start
    (2) Repeat while p<>NULL
            If(p⇒next = NULL)
                DeleteNode(p)
```



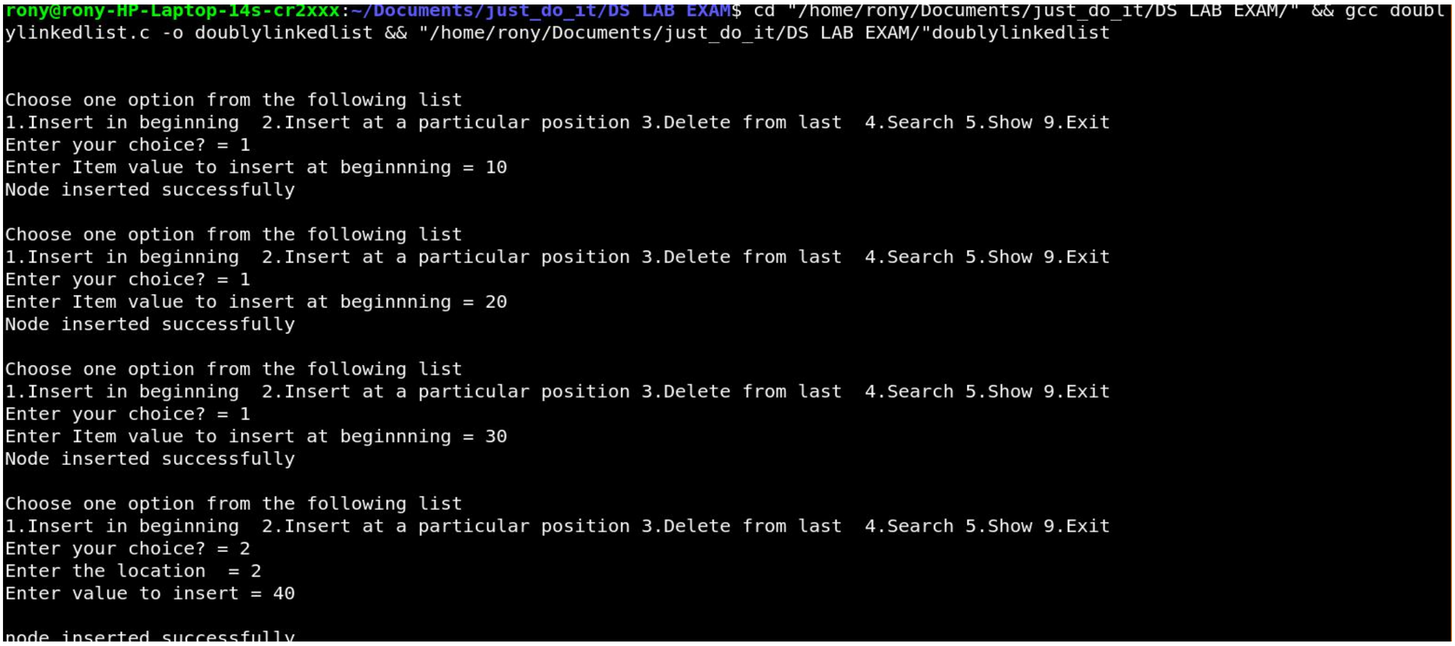
```
(4) Display
    (1) p = start
    (2) Repeat while p<> NULL
            print p ⇒data

(5) Search
    (1) input the item which want to search as data
    (2) P = start
    (3) Repeat while p<> Null
            If (p ⇒ info = data)
                print the location of the node.
```

Program:

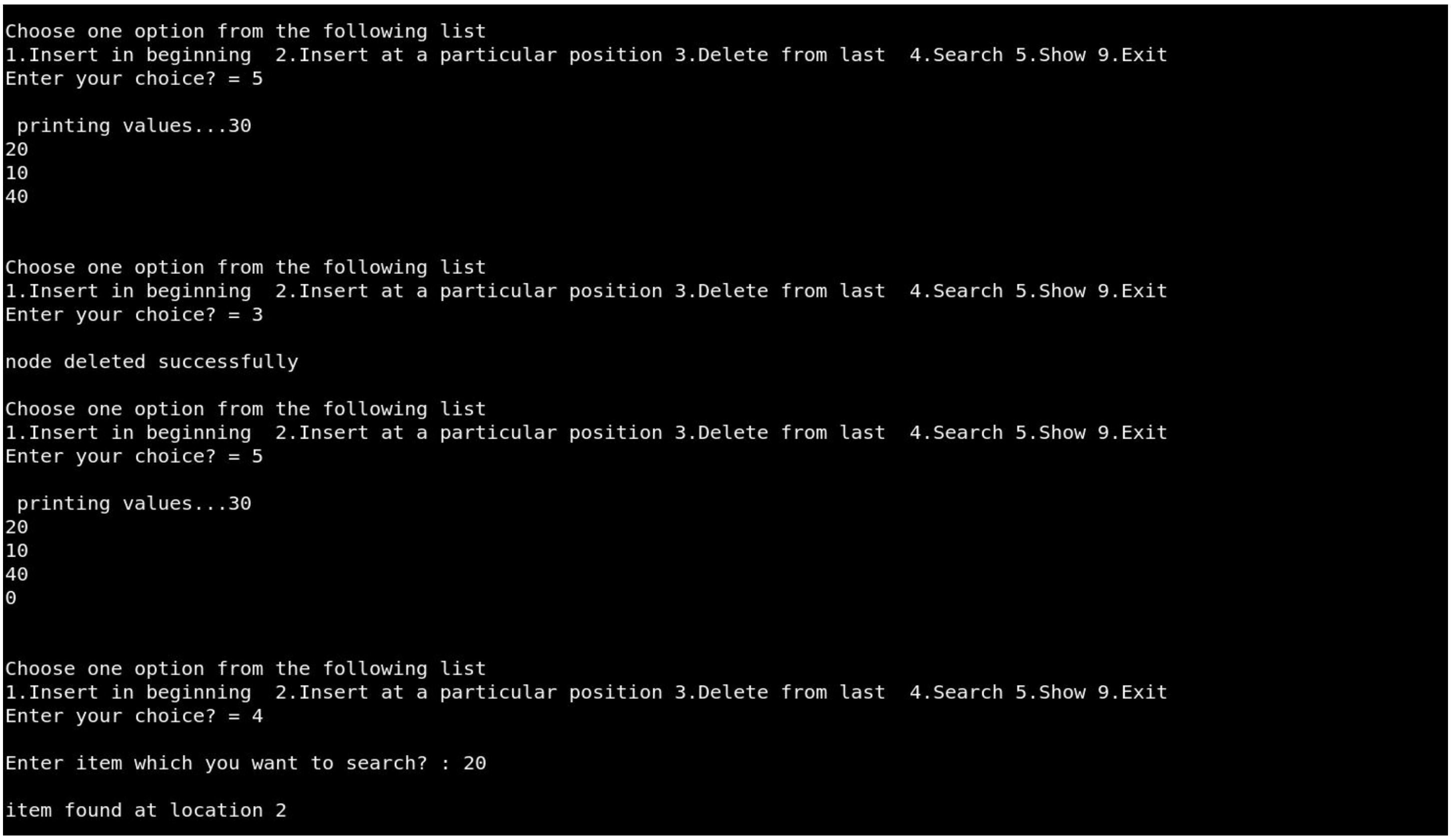
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insert_at_beginning();
void insert_at_specified();
void deletion_at_last();
void display();
void search();
void main ()
{
int choice =0;
    while(choice != 9)
    {
        printf("\n");
        printf("\nChoose one option from the following list");
        printf("\n1.Insert in beginning  2.Insert at a particular
position 3.Delete from last  4.Search 5.Show 9.Exit");
        printf("\nEnter your choice? = ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insert_at_beginning();
            break;
            case 2:
            insert_at_specified();
            break;
            case 3:
            deletion_at_last();
            break;
            case 4:
            search();
            break;
            case 5:
            display();
            break;
            case 6:
            exit(0);
            break;
            default:
            printf("Please enter valid choice in the menu");
        }
    }
}
void insert_at_beginning()
{
   struct node *ptr;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("Enter Item value to insert at beginnning = ");
    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("Node inserted successfully");
}
```

```
}

void insert_at_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 in DLL",
loc);
               return;
           }
       }
       printf("Enter value to insert = ");
       scanf("%d",&item);
       ptr->data = item;
       ptr->next = temp->next;
       ptr -> prev = temp;
       temp->next = ptr;
       temp->next->prev=ptr;
       printf("\nnode inserted successfully\n");
   }
}

void deletion_at_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted successfully");
    }
    else
    {
        ptr = head;
        while(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nnode deleted successfully");
    }
}

void display()
{
    struct node *ptr;
    printf("\n printing values...");
    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("\nEmpty List");
    }
    else
    {
        printf("\nEnter item which you want to search? : ");
        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");
        }
    }     }
```

Result:



```
rony@rony-HP-Laptop-14s-cr2xxx:~/Documents/just_do_it/DS LAB EXAM$ cd "/home/rony/Documents/just_do_it/DS LAB EXAM/" && gcc doubl
ylinkedlist.c -o doublylinkedlist && "/home/rony/Documents/just_do_it/DS LAB EXAM/"doublylinkedlist


Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 1
Enter Item value to insert at beginnning = 10
Node inserted successfully

Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 1
Enter Item value to insert at beginnning = 20
Node inserted successfully

Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 1
Enter Item value to insert at beginnning = 30
Node inserted successfully

Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 2
Enter the location  = 2
Enter value to insert = 40

node inserted successfully
```



```
Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 5

 printing values...30
20
10
40


Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 3

node deleted successfully

Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 5

 printing values...30
20
10
40
0


Choose one option from the following list
1.Insert in beginning  2.Insert at a particular position 3.Delete from last  4.Search 5.Show 9.Exit
Enter your choice? = 4

Enter item which you want to search? : 20

item found at location 2
```