1st Semester Lab Exam

Data Structures

Submitted By

Mohammed Rasik

Roll No. 223

RegNo. TKM20MCA-2023

GIT LINK : https://github.com/RazikMhd/s1LabExam/tree/master/datastructure

Question

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

Assume that START is the first element in the linked list and TAIL is the last element of linked list.

i. Insert At Beginning

1. Start
2. Input the DATA to be inserted
3. Create a new node.
4. NewNode → Data = DATA NewNode →Lpoint =NULL
5. IF START IS NULL NewNode→ Rpoint = NULL
6. Else NewNode → Rpoint = START START→Lpoint = NewNode
7. START =NewNode
8. Stop

ii. Insertion at location:

1. Start
2. Input the DATA and POS
3. Initialize TEMP = START; i = 0
4. Repeat the step 4 if (i less than POS) and (TEMP is not equal to NULL)
5. TEMP = TEMP → RPoint; i = i +1
6. If (TEMP not equal to NULL) and (i equal to POS)

(a) Create a New Node

(b) NewNode → DATA = DATA

(c) NewNode → RPoint = TEMP → RPoint

(d) NewNode → LPoint = TEMP

(e) (TEMP → RPoint) → LPoint = NewNode

1. (f) TEMP → RPoint = New Node
2. Else

(a) Display “Position NOT found”

1. Stop

iii. Insert at End

1. Start
2. Input DATA to be inserted
3. Create a NewNode
4. NewNode → DATA = DATA
5. NewNode → RPoint = NULL
6. If (START equal to NULL)

a. START = NewNode

b. NewNode → LPoint=NULL

1. Else

a. TEMP = START

b. While (TEMP → Next not equal to NULL)

i. TEMP = TEMP → Next

c. TEMP → RPoint = NewNode

d. NewNode → LPoint = TEMP

1. Stop

iv. Forward Traversal

1. Start
2. If (START is equal to NULL)

a) Display “The list is Empty”

b) Stop

1. Initialize TEMP = START
2. Repeat the step 5 and 6 until (TEMP == NULL)

3. Display “TEMP → DATA”
4. TEMP = TEMP → Next
5. Stop

v. Backward Traversal

1. Start
2. If (START is equal to NULL)
3. Display “The list is Empty”
4. Stop
5. Initialize TEMP = TAIL
6. Repeat the step 5 and 6 until (TEMP == NULL)
7. Display “TEMP → DATA”
8. TEMP = TEMP → Prev
9. Stop

Algorithm

2. Doubly linked list

1. Insert at beginning

1. Start
2. input data to be inserted
3. Create new node
4. NewNode → data = data newNode → Lpoint = NULL
5. If start is NULL NewNode → Rpoint = NULL
6. else newnode → Rpoint = start start → Lpoint = NewNode
7. Start = NewNode
8. Stop.

2. Insert at location

12 1. Start
2. Input the Data & POS.
3. Initialize TEMP = START, i = 0
4. Repeat the step 4 if (less than POS)
5. TEMP = TEMP → R point; i = i + 1
6. If (TEMP not equal to NULL) & (i equal to POS)

3. Insert at End :

1. Start
2. input DATA to be inserted
3. Create newnode
4. new Node → data = Data
5. New nod → R point = NULL
6. if (start equal to NULL

4. forward traversal

1. start
2. if (start equal to NULL).

5. Backward traversal

1. Start.
2. if (start is = to NULL)
3. display "Empty list."
4. stop.
5. initialize TEMP = TAIL.
6. Repeat the Step 5 & 6 until (TEMP is NULL)
7. Display "TEMP → DATA".
8. TEMP = TEMP → Prev
9. Stop.

CODE :

GIT LINK :
https://github.com/RazikMhd/s1LabExam/blob/master/datastructure/doubly_linked_list.c

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insert_beginning();
void insert_last();
void insert_specified();
void delete_beginning();
void delete_last();
void delete_specified();
void display();
void search();
void main ()
{
int choice =0;
    while(choice != 9)
    {
        printf("\n*********Program Menu*********");
        printf("\nChoose an Operation");
        printf("\n1.Insert in begining 2.Insert at last 3.Insert at any random
 location 4.Delete from Beginning 5.Delete from last 6.Delete the node after t
he given data 7.Search 8.Show 9.Exit");
        printf("\nEnter your choice? = ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insert_beginning();
            break;
            case 2:
                    insert_last();
            break;
            case 3:
            insert_specified();
            break;
            case 4:
            delete_beginning();
            break;
```

```
            case 5:
            delete_last();
            break;
            case 6:
            delete_specified();
            break;
            case 7:
            search();
            break;
            case 8:
            display();
            break;
            case 9:
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void insert_beginning()
{
   struct node *ptr;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("Enter 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("Node inserted");
}

}
void insert_last()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("OVERFLOW");
   }
   else
   {
       printf(" Enter 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("node inserted");
    }
void insert_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 delete_beginning()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted");
    }
    else
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nnode deleted");
    }

}
```

```c
void delete_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted");
    }
    else
    {
        ptr = head;
        while(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nnode deleted");
    }
}
void delete_specified()
{
    struct node *ptr, *temp;
    int val;
    printf("\n Enter the data after which the node is to be deleted : ");
    scanf("%d", &val);
    ptr = head;
    while(ptr -> data != val)
    ptr = ptr -> next;
    if(ptr -> next == NULL)
    {
        printf("\nCan't delete");
    }
    else if(ptr -> next -> next == NULL)
    {
        ptr ->next = NULL;
    }
    else
    {
        temp = ptr -> next;
        ptr -> next = temp -> next;
        temp -> next -> prev = ptr;
        free(temp);
```

```
        printf("\nnode deleted");
    }
}
void display()
{
    struct node *ptr;
    printf("\n printing nodes...");
    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");
        }
    }
```

```
}
```

OPERATION :

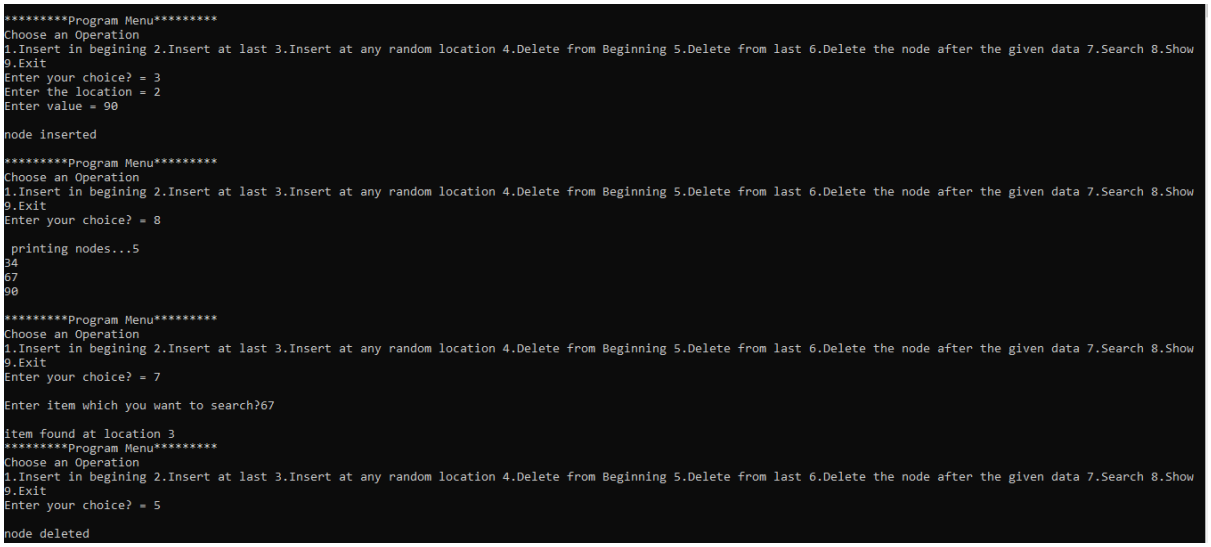
Made a doubly linked list with 3 nodes : 5,34,67.

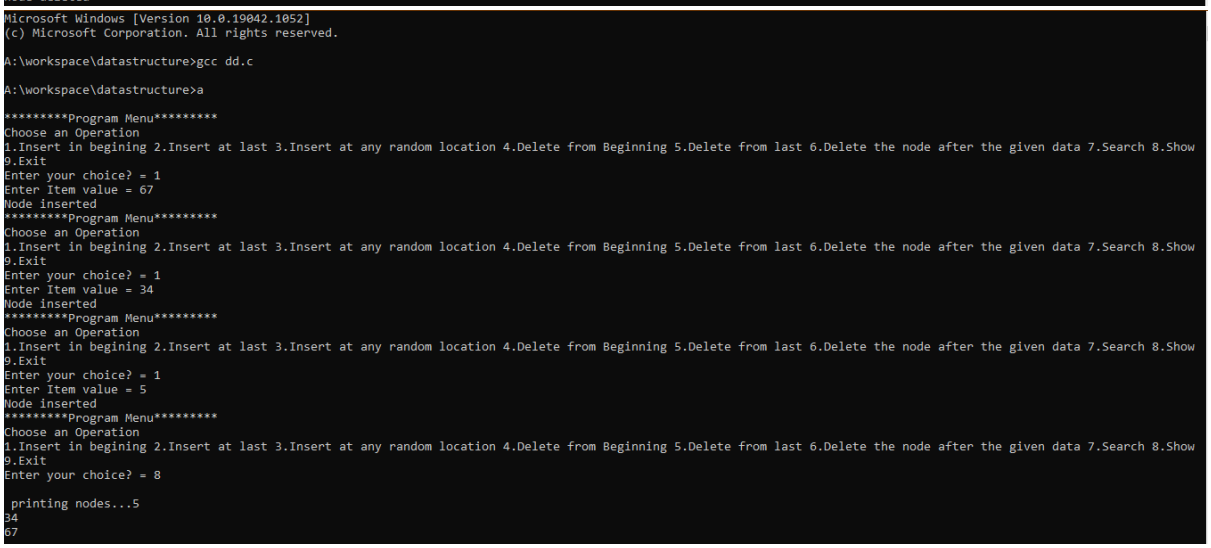
A) Inserted 90 at position 2.

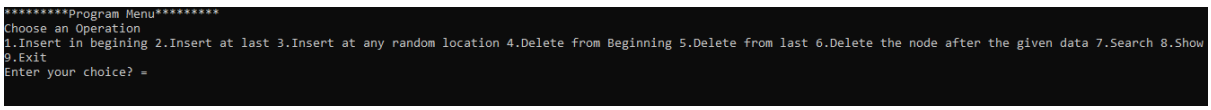
B) Found element 67 at position 3

c) Deleted element 67.

OUTPUT







OUTPUT (TEXT):

Microsoft Windows [Version 10.0.19042.1052]

(c) Microsoft Corporation. All rights reserved.

A:\workspace\datastructure>gcc dd.c

A:\workspace\datastructure>a

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 1

Enter Item value = 67

Node inserted

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 1

Enter Item value = 34

Node inserted

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 1

Enter Item value = 5

Node inserted

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 8

printing nodes...5

34

67

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 3

Enter the location = 2

Enter value = 90

node inserted

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 8

printing nodes...5

34

67

90

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 7

Enter item which you want to search?67

item found at location 3

*********Program Menu*********

Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? = 5

node deleted

*********Program Menu*********

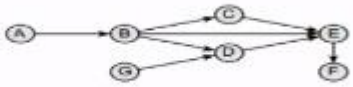
Choose an Operation

1.Insert in begining 2.Insert at last 3.Insert at any random location 4.Delete from Beginning 5.Delete from last 6.Delete the node after the given data 7.Search 8.Show 9.Exit

Enter your choice? =

Question

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



Develop a program to implement topological sorting.

ALGORITHM

1) Identify a node with no incoming edges.

2) Add that node to the ordering.

3) Remove it from the graph.

4) Repeat

Algorithms

Q. Topological Sorting

1. ~~Identify a node with no incoming edges.~~
2. ~~Add that node to the ordering~~

1. Start.
2. Create char array with vertices of ~~as~~ acyclic path graph.
   { "A", "B", "C" ... "Z" }
3. Input adjacency matrix as a 2-D array.
4. Set all elements.
5. Calculate the vertices which has in-degree 0 & decrease th in-degree count of vertices who are adjacent to the vertex.
6. Add that node to the ordering & remove it from graph.
7. Repeat step 6 until all nodes are removed.

8. print the ordering with respect to the char array.

CODE :

GIT LINK :
https://github.com/RazikMhd/s1LabExam/blob/master/datastructure/topolical_sorting.c

```c
#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','G', 'H', 'I', 'J', 'K', 'L', 'M'
, 'N', 'O'};

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

printf("Enter the adjacency matrix values:\n");
for(i=0;i<n;i++){
printf("Enter data for 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;
}
```

OPERATION :

Input the following matrix :

0100000

0011100

0000100

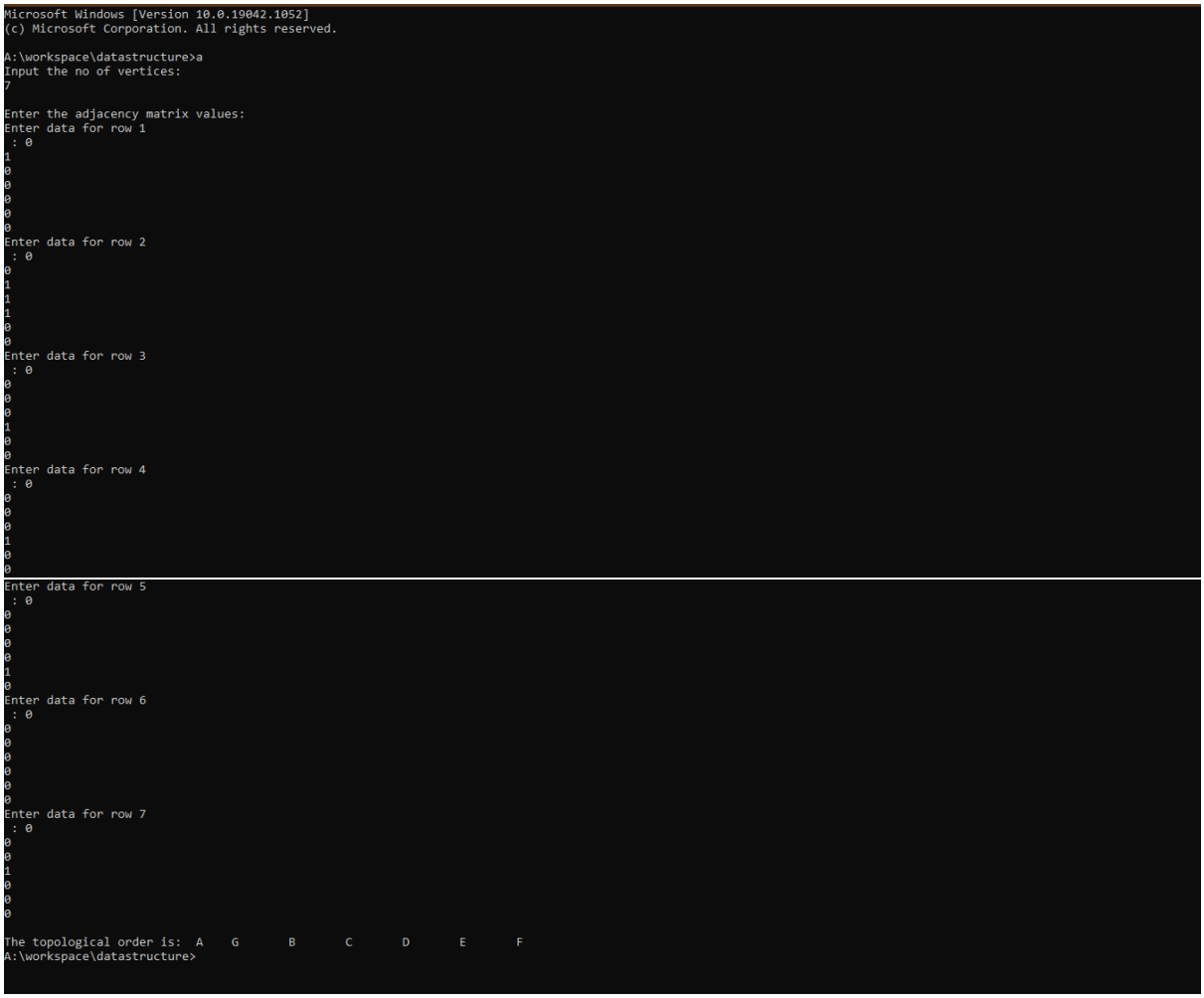
0000100

0000010

0000000

0001000

Output



OUTPUT TEXT

Microsoft Windows [Version 10.0.19042.1052]

(c) Microsoft Corporation. All rights reserved.

A:\workspace\datastructure>a

Input the no of vertices:

7

Enter the adjacency matrix values:

Enter data for row 1

: 0

1

0

0

0

0

0

Enter data for row 2

: 0

0

1

1

1

0

0

Enter data for row 3

: 0

0

0

0

1

0

0

Enter data for row 4

: 0

0

0

0

1

0

0

Enter data for row 5

: 0

0

0

0

0

1

0

Enter data for row 6

: 0

0

0

0

0

0

0

Enter data for row 7

: 0

0

0

1

0

0

0

The topological order is: A G B C D E F