DATA STRUCTURE LAB EXAM

Jomin K Mathew

TKM20MCA2021

Roll.no: 20MCA221

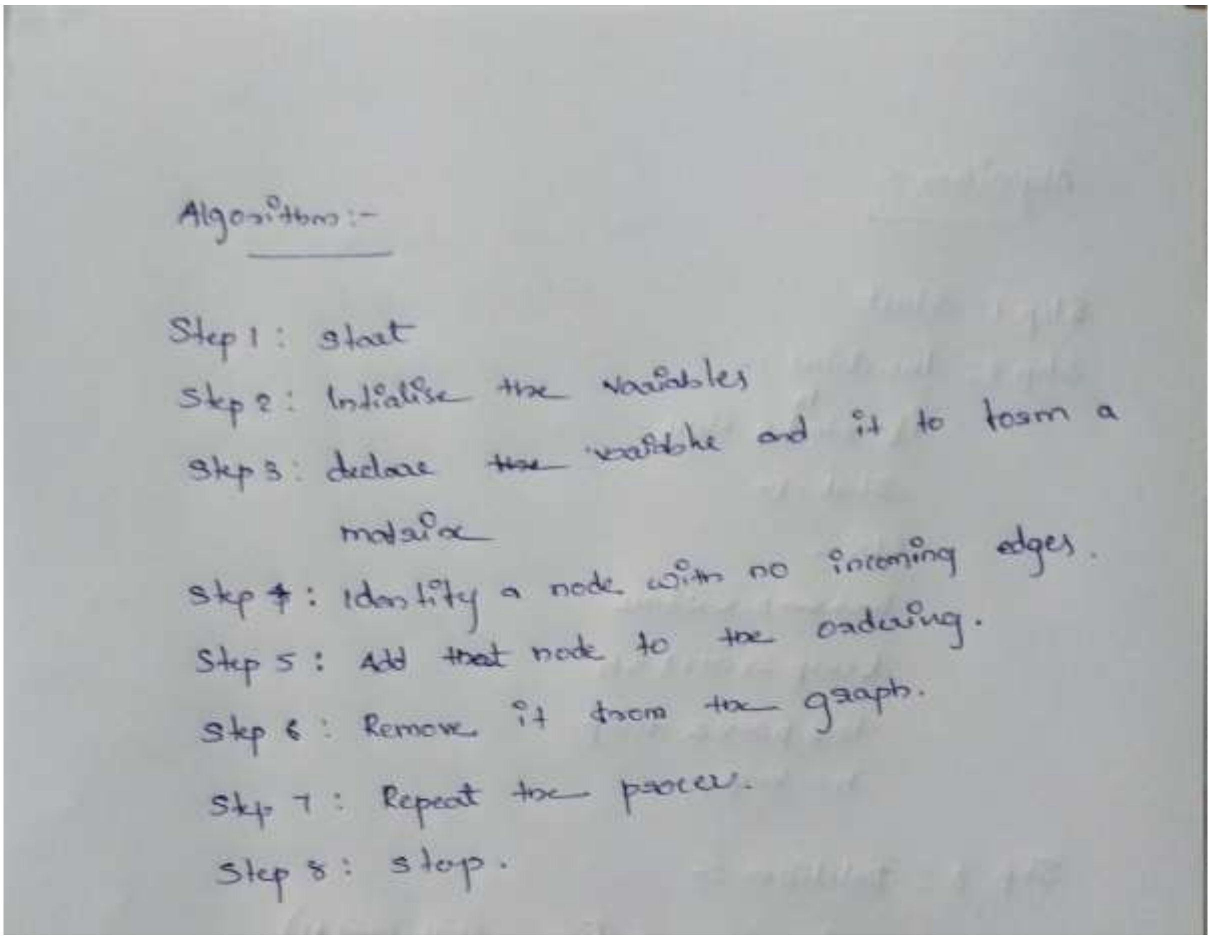
Git Link: https://github.com/jominkmathew/Data-Structure/blob/master/Data%20structure%20Lab%20Exam/TKM20MCA-2021-Jomin%20K%20Mathew.pdf

Question 1:

Consider a directed acyclic graph:

Develop a program to implement topological sorting

Algorithm:



Algorithm :-

Step 1 : start

Step 2 : Initialise the variables

Step 3 : declare the variable and it to form a matrix

Step 4 : Identify a node with no incoming edges.

Step 5 : Add that node to the ordering.

Step 6 : Remove it from the graph.

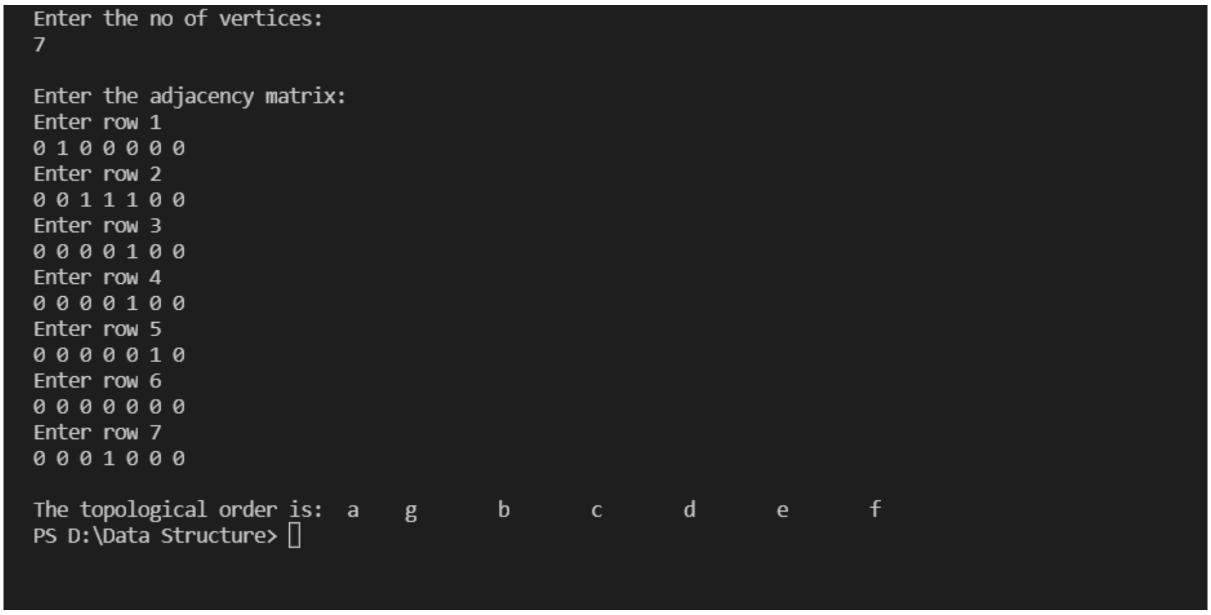
Step 7 : Repeat the process.

Step 8 : stop.

Program:

```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' };

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;
}
```

Output:



```
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
PS D:\Data Structure> 
```

Question 2:

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 :-

Step1 : Start

Step2 : Declaring the position where to be inserted on Adde functions using switch case.

Step3 : Insertion :-

```
if (h == NULL)
    create();
    h = temp;
    temp1 = h;
else
    temp -> next = h;
    h -> prev = temp;
    h = temp;
Insertion at end ;
    if (h == NULL)
        h = temp
        temp1 = h;
    else
        temp1 -> next = temp;
        temp -> prev = temp1;
        temp1 = temp;

Insert at any ;
    if ((pos < 1) (pos >= count + 1)
    if (h == NULL) && (pos != 1))
        if (h == NULL) && (pos (== 1))
```

```
h = temp ;
temp = h ;
else
  while (i < pos)
    temp2 = temp2 -> next ;
    i++ ;
  temp -> prev = temp2 ;
  temp -> next = temp2 -> next
  temp2 -> next -> prev = temp
  temp2 -> next = temp ;

Step 4 : deletion :-
    if (pos <1) (pos >= (count+1))
    if (h == NULL)
    else
      while (i<pos)
      temp2 = temp2 -> next ;
      i++
    if (temp2 -> next = = NULL)
        temp2 = h = NULL
    if (temp2 -> next = = NULL)
        temp2 -> next prev = temp2 -> prev
    if ( i != 1)
        temp2 -> prev -> next = temp2 -> next;
```

Step 5: Search:-

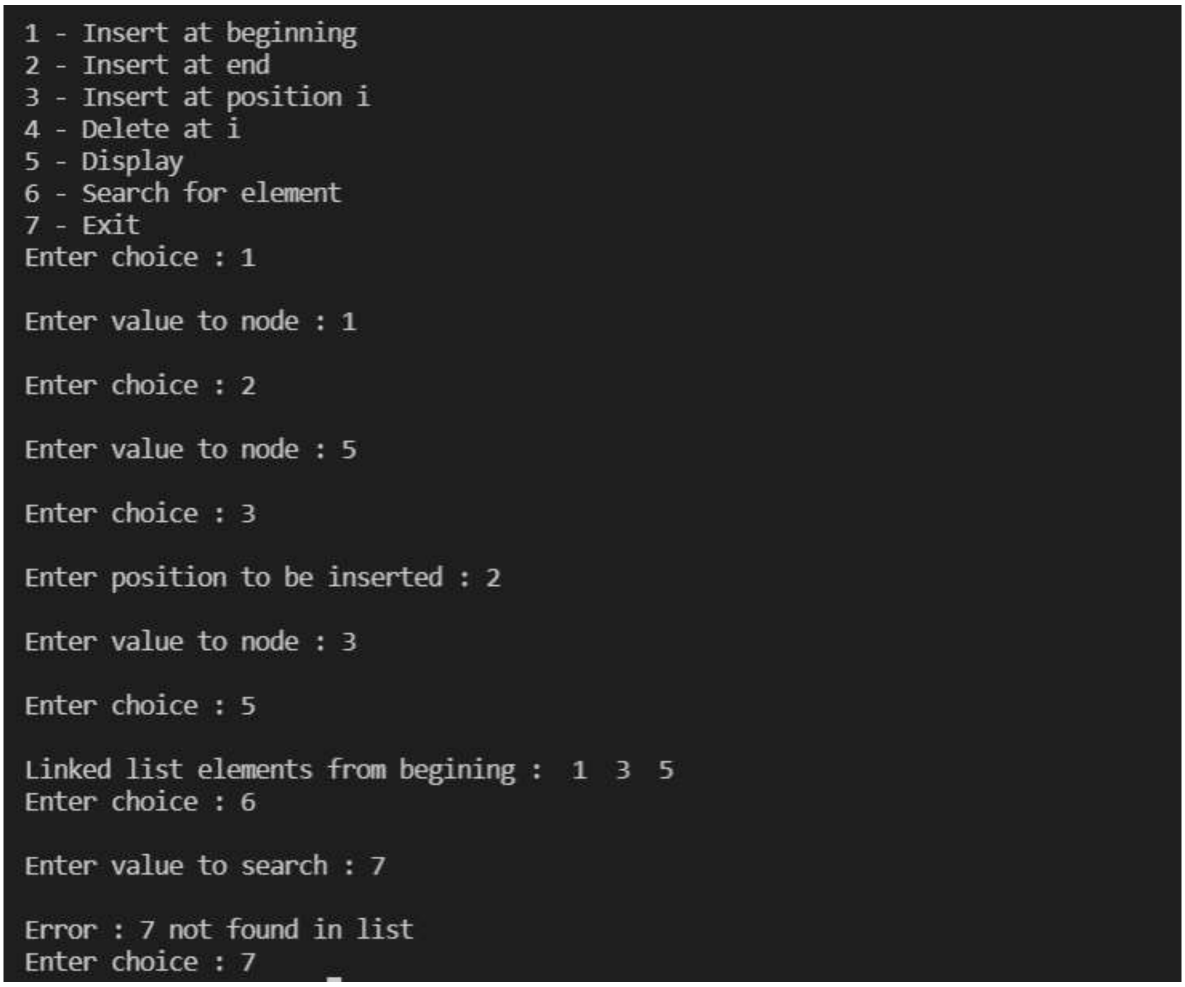
```
        if temp = = NULL
        while (temp2 != NULL)
            if (temp2 -> n = = data)
          else
              temp2 = temp2 -> next;
              count++
```

Step 6: Stop.

Program:

```c
#include <stdio.h>
#include <stdlib.h>

struct node
{
    struct node *prev;
    int n;
    struct node *next;
}*h,*temp,*temp1,*temp2,*temp4;

void insert1();
void insert2();
void insert3();
void traversebeg();
void traverseend(int);
void sort();
void search();
void update();
void delete();

int count = 0;

void main()
{
    int ch;

    h = NULL;
    temp = temp1 = NULL;

    printf("\n 1 - Insert at beginning");
    printf("\n 2 - Insert at end");
    printf("\n 3 - Insert at position i");
    printf("\n 4 - Delete at i");
    printf("\n 5 - Display");
    printf("\n 6 - Search for element");
    printf("\n 7 - Exit");

    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            insert1();
            break;
        case 2:
            insert2();
            break;
        case 3:
            insert3();
            break;
        case 4:
            delete();
            break;
        case 5:
            traversebeg();
            break;
        case 6:
            search();
            break;
        case 7:
            exit(0);
        default:
            printf("\n Wrong choice menu");
        }
    }
}

void create()
{
    int data;

    temp =(struct node *)malloc(1*sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter value to node : ");
    scanf("%d", &data);
    temp->n = data;
    count++;
}

void insert1()
{
    if (h == NULL)
    {
        create();
        h = temp;
        temp1 = h;
    }
    else
    {
        create();
        temp->next = h;
        h->prev = temp;
        h = temp;
    }
}


void insert2()
{
    if (h == NULL)
    {
        create();
        h = temp;
        temp1 = h;
    }
    else
    {
        create();
        temp1->next = temp;
        temp->prev = temp1;
        temp1 = temp;
    }
}

void insert3()
{
    int pos, i = 2;

    printf("\n Enter position to be inserted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Position out of range to insert");
        return;
    }
    if ((h == NULL) && (pos != 1))
    {
        printf("\n Empty list cannot insert other than 1st position");
        return;
    }
    if ((h == NULL) && (pos == 1))
    {
        create();
        h = temp;
        temp1 = h;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        create();
        temp->prev = temp2;
        temp->next = temp2->next;
        temp2->next->prev = temp;
        temp2->next = temp;
    }
}

void delete()
{
    int i = 1, pos;

    printf("\n Enter position to be deleted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Error : Position out of range to delete");
        return;
    }
    if (h == NULL)
    {
        printf("\n Error : Empty list no elements to delete");
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf("Node deleted from list");
                free(temp2);
                temp2 = h = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf("Node deleted from list");
            return;
        }
        temp2->next->prev = temp2->prev;
        if (i != 1)
            temp2->prev->next = temp2-
>next;    /* Might not need this statement if i == 1 check */
        if (i == 1)
            h = temp2->next;
        printf("\n Node deleted");
        free(temp2);
    }
    count--;
}

void traversebeg()
{
    temp2 = h;

    if (temp2 == NULL)
    {
        printf("List empty to display \n");
        return;
    }
    printf("\n Linked list elements from begining : ");

    while (temp2->next != NULL)
    {
        printf(" %d ", temp2->n);
        temp2 = temp2->next;
    }
    printf(" %d ", temp2->n);
}
void search()
{
    int data, count = 0;
    temp2 = h;

    if (temp2 == NULL)
    {
        printf("\n Error : List empty to search for data");
        return;
    }
    printf("\n Enter value to search : ");
    scanf("%d", &data);
    while (temp2 != NULL)
    {
        if (temp2->n == data)
        {
            printf("\n Data found in %d position",count + 1);
            return;
        }
        else
             temp2 = temp2->next;
            count++;
    }
    printf("\n Error : %d not found in list", data);
}
```

Output:



```
1 - Insert at beginning
2 - Insert at end
3 - Insert at position i
4 - Delete at i
5 - Display
6 - Search for element
7 - Exit
Enter choice : 1

Enter value to node : 1

Enter choice : 2

Enter value to node : 5

Enter choice : 3

Enter position to be inserted : 2

Enter value to node : 3

Enter choice : 5

Linked list elements from begining :  1  3  5
Enter choice : 6

Enter value to search : 7

Error : 7 not found in list
Enter choice : 7
```