

## LABORATORY WORK BOOK

Janzan zahaib

Name of the Student						Roll Number						
Class CSE-C Semester I							3 1	9 5	IA	0 5	5 +	7
Cou	rse Cod	e ACSDOG Course	Name	ps Lab		2				757	-	
Nan	ne of the	Course Faculty D.S. M.	Madhus	adhan	Redd	I		Fa	culty ID :	JAKE.	10%7	1.1
		mber: 0.6	Week Number 06			Date						
	Exercise Number		MARKS AWARDED									
			Aim/ Preparation	Algorithm / Procedure Performance in the Lab		Source Code Calculations and Graphs		-	Program Execution Results and Error		Tota	Total
								Analysis		-		
			4			4	-	4	4	20	-	
1	5-1	Build a Graph			193 Lm			1	1.40			
2	5-2	Number of sink nades in a graph						12	-1/1			
3	5.3	connected components in a Grouph		4	C Terrori	L	(		4	4	20	>
4	5.4	Transpose graph		18	CT#	Phi	1.1	->	ast H			
5	5.5	counting telplets				4-5	074	P.Y.				
6						6		-30				
7										1-9		
8												
9												
10		1-11-11-11	1									
11										Hosp		
12		Committee Park										

Signature of the Student

Signature of the Faculty

## START WRITING FROM HERE

```
5-2 Number of sink nodes in a graph :-
 mv = int (input())
me = int (input())
 adj_list = E vH: [] for v in Hamge (n)}
 edges = []
 for i in Hange (ne):
       edges = tuple (map (int, input () split ()))
     edges. append (edge)
 for v1, v2 in edges:
      adj_list (V1] append (V2)
 sink_nodes = []
for k, v in adj_list. Items ():
     if len (adj_list LKJ) = = 0:
           sink-nodes append (k)
prink (lem (sink nodes).
Input:
3 2
output :-
3
```

```
connected components in a Greath.
5.3
dfs. det dfs (ventex, visited):
          visited [verter-1] - Toue
          for i in adj_list (vertex ):
              if not visited [i-1]:
                  dts (i, visited)
nv, ne = list(map(int, input (), split()))
visited = Etalse for i in stampe (nv)3
adj_list = Ev+1: [] for v in range (nv)}
edges = []
for i in stange (nc):
      edge = tuple Cmap (int. input (). split ()))
      edges = append (edge).
for vivve in edges. In a land it is possible to the
     adj-list (VI) append (V2)
      adj-list [v2] append [v]
component = 0
for i in range (1, 1741);
if visited [i-1] = = false
     component + = 1
      dfs (i, visited)
     print (component).
```

```
input:
   output :-
   3
J.4 Transpose Croaph
   nv = int (input())
   adj_list = [v:[] for v in Hange (nv)}
   for i in Harnge (nv);
   edge = tuple (map (bt &, input (). split (': ')))
        L = int (edge [0])
        V - []
        for i in edge [1]:
            if J. is digit();
                v. append (int (i))
            adj-list [k] =V
    +rons_adj_list - Ev:[] for vin Hange (n)}
   for k, v in adj_list_items ();
```

```
for i in v:
                                    for viva in edges:
          toans-adj_list [i] append (1)
for 1c, v In toans adjulist . items (DEN) (N) for you
          print (K, 1:1, V) (i) boggo. (ii) leil ibo
                               (p) brogge . Evel toil - the
Input:
                                   print ('Adjocency rendriv')
                                       stom ibo of i rob
0 : 1
                                   : [1-1] 7 mi i cot
1;2
                            print (i, end. ', ')
2 13
                                         ([1-]1) this q
3 :
output :-
                                    C' tais ( Adjacency list )
                               :() ensti. tell-(bo ni vei vot
0:13
                     paint (ii' souted (list (set (v))))
1:[0]
2:[1]
3:[2]
      Reprensitation of a graph:
                                                       LO
5.1
                                                       2.1
nv, ne = list (map (int input (). split (', ')))
```

sol Repsenstation of a graph:

nv, ne = list (map (Int input (). split (', ')))

edges = []

for i in stange (ne):

edges = tuple (map (int, input (). split (', ')))

edges. append (edge)

adj-Inat = [[o] "nv tor i" in stange (nv)]

adj-list = [[v] : [] for v in stange (nv)]

```
for vi, vi in edges:
      adj_mat (v,) (v,) = 1
      adj - mat [v,] [v,] = 1
     adi_list [vi]. appord (v2)
     adj-list [v2]. append (4)
print ( 'Adjacency matrix')
for i in adi_mat:
     foo i in i [:-1]:
         print (i, erd = ', ')
     print (1[-1])
print ('Adjacency list')
for i,v in adj-list itens ():
     print (i, : , sorted (list (set (v))))
Input:
3, 3
0,1
1,2
           my ne - list (map (Int. input (), split (' ' )))
2,0
output
Adjacency matrix
edges - tuple (map (not, input (), split (1, 1 )), 1, 0
                            edges, append (edge)
1,10
Adjacency List (Como agrana di 1 rat un 1000 - harrilla
0: [1,2]
                Elvas sparon of u rot 13 = 431 - fell ite
1: [0,2]
 25 [0, 1].
```

```
5.5
N, M = list(map Cint, input (), split()))
edges = []
for i in stampe (ne):
    edge = tuple (map (int, input (). split (', ')))
     edges append (edge)
adj_mat =
wedges = []
for i in range (1, N);
    too j in Hange (i+1, N+1):
        计门门:
       if (i, i) not in edges:
                wedgesappend (6,1)
triedges = 13
for in in stange (1, N+1);
   for j in Hange (i+1, N+1);
       for k in stange (j+1, N+1):
             if ((i,i) in edges) and ((i, k) in edges) and ((i, k in
                 triedges oppend ((i,i, 10)
             elif (Ci,i) in wedges) and (Ci, 12) in wedges and
                                          ((i, ie in wedge):
                 tokedges - append((i, jo 10))
print (1em (triedges))
```