CSE-221

Assignment 1+2

Answer to the Question.1

1. In this example the edge is going from node A to node B and It is a directed graph therefore. A is adjacet to B but B is not adjacent to A because we can not go to A from node B.

$$(A) \longrightarrow (B)$$

2. If there is n amount of vertices we can create $^{n}C_2$ or $\frac{n(n-1)}{2}$ number of edges in an undirected graph ~~with~~ Without creating allowing loops and multiple edges between pairs.

3. In case of directed graph :- $n(n-1)$.

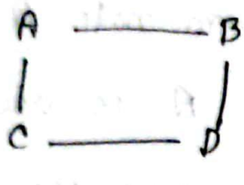
4. We can construct a $(n-1)$ edges.

5. For directed graph we can constuct $(n-1)$ edges avoiding cycle.

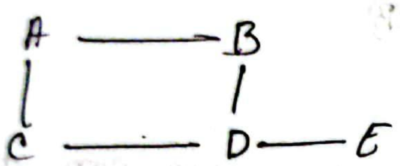
6. The property is Eulerian ~~property~~ Path. Where every edge is visited excty onle.

7. The property is Hamiltonian path where every node is visited exacty once and some edges may remain unvisited.

Eulenian :-



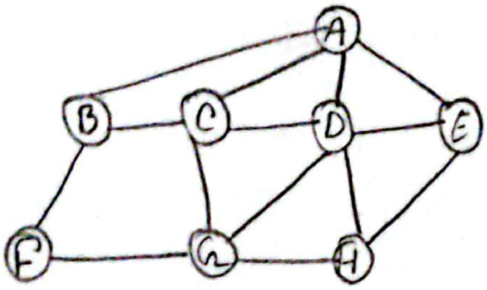
Hamiltotonian :-



8) Addacency list is usually more space efficient compared to addacency matrix. For example addececny matrix uses $O(V^2)$ sapce of andirected and sparse graph while addececny list uses $O(V+E)$ space for those graph. Similarly in weighted, unweighted, Dense graph addececny list uses less memony compared to addececny matrix.

Answen to the Question. 2

A



B

Addaceney list:-

0 → A → B → C → D → E

1 → B → A → C → F

2 → C → A → B → D → G

3 → D → A → C → E → G → H

4 → E → A → D → H

5 → F → B → G

6 → G → C → D → F → H

7 → H → D → E → G

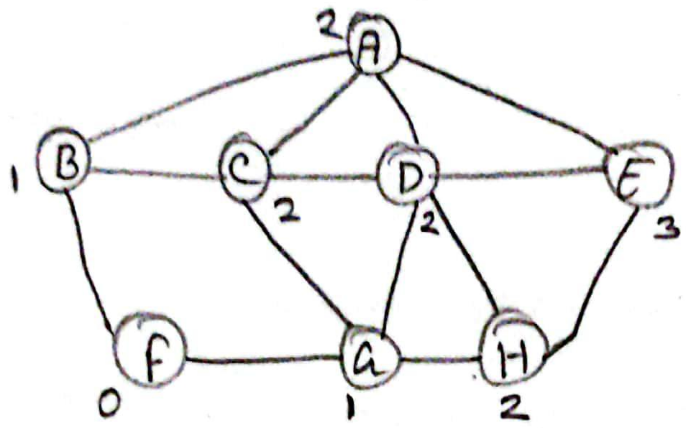
Adjacency Matrix :-

	A	B	C	D	E	F	G	H
A	0	1	1	1	1	0	0	0
B	1	0	1	0	0	1	0	0
C	1	1	0	1	0	0	1	0
D	1	0	1	0	1	0	1	1
E	1	0	0	1	0	0	0	1
F	0	1	0	0	0	0	1	0
G	0	0	1	1	0	1	0	1
H	0	0	0	1	1	0	1	0

C

AB - 1
AC - 2
AD - 2
AE - 1
AF - 1
AG - 2
AH - 2
BC - 1
BD - 2
BE - 1
BF - 0
BG - 2
BH - 0
CD - 2
CE - 2
CF - 2
CG - 1
CH - 2
DE - 2
DF - 1
DG - 2
DH - 2
EF - 0
EG - 2
EH - 1
FG - 0
FH - 1
GH - 1

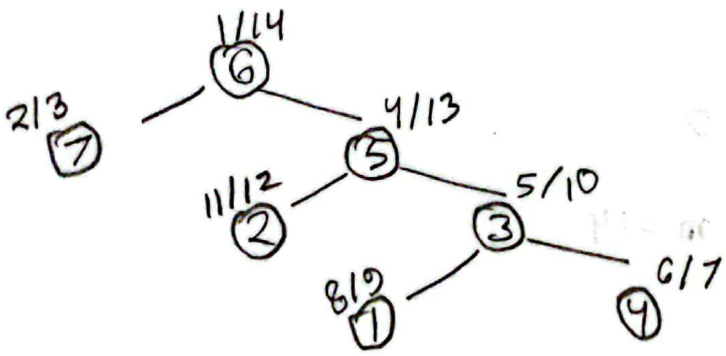
D



Here If conside F as the root E is 3 degree connection away from it therefore F will not be able to see E's posts.

Answer to the Question. 3

a



b

Nodes	1	2	3	4	5	6	7
Parent	3	5	5	3	6	-1	6
Starting time	8	11	5	6	4	1	2
finish time	9	12	10	7	13	14	3
Distance from Root	3	2	2	3	1	0	1

Answer to the Question. 4

A

Here,

Vertex $= 9$

edge, $m = 14$

Now,

degree of,

$A = 3$	$E = 3$
$B = 4$	$F = 3$
$C = 4$	$G = 4$
$D = 3$	$H = 2$
	$S = 2$

$\therefore$ total Degree $= (3+4+4+3+3+3+4+2+2)$

$= 28$

$\therefore \sum \deg = 28 = 2 \times m \quad | \; m = 14$

[Proved]

B

Maximum of edges possible in this undirected graph $= \frac{n(n-1)}{2}$

$$= \frac{9(9-1)}{2}$$

$$= \frac{72}{2}$$

$$= 36$$

$\therefore$ We can add more $(36-14)$ edges

$= 22$ edges.

Answer to the Question. 5

A

DFS (Socks)
↓
DFS (Nagra)
↓
DFS (Watch)
↓
DFS (Engagement ring)
↓
DFS (Glasses)

DFS (Turban)
↓
DFS (Broach)

DFS (Undergarment)
→ DFS (Pajama)
→ DFS (Shenwani)

Undergarments
Shenwani
Pajama
Turban
Broach
Socks
Nagra
Watch
Engagement ring
Glasses

His dressing order should be:-

Undergarments → Sherwani → Pajamas → turban → broach → socks → Nagra → watch → Engagement ring → Glasses.

D

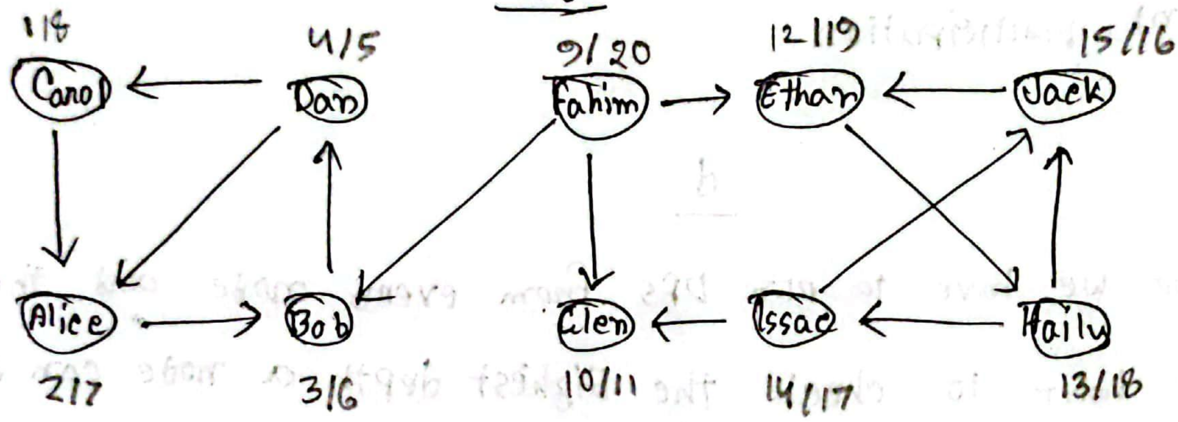
No all of my classmets will not have the same ordering because we can all start from ~~b~~ differento point but the every obdect will come after some specific ~~ob~~ obectes. for example Nagra will always come after socks event if the start from any other point.

Answer to the Question. 6

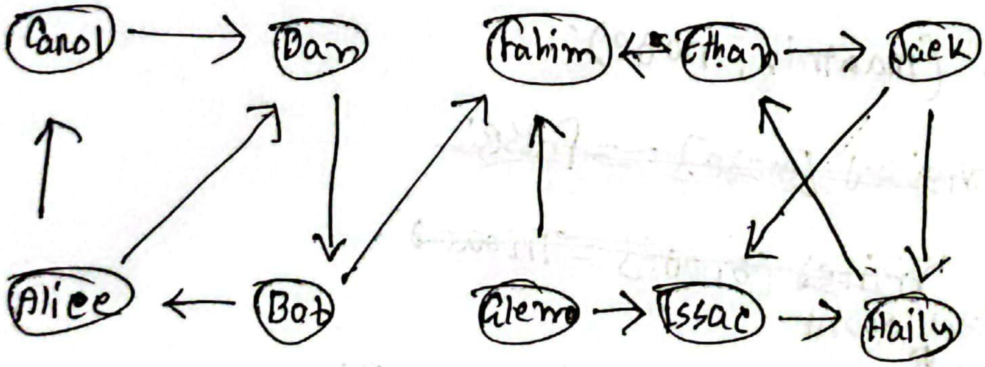
a

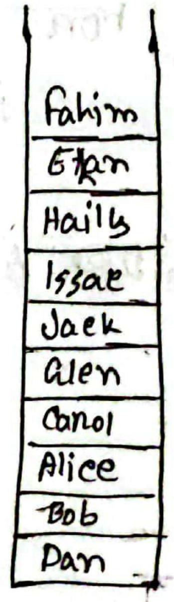
The algorithm tailored for this situation is kasaraju algorithm.

b



reverse:-





stack

~~Group 1: fahim~~

Group 01:- ~~Ethan, Jack~~ Haily, Issac, Jack, Ethan

~~Group 02: Glen~~

Group 02:- Alice, Bob, Dan, carol

∴ 2 groups can be formed.

c

Fahim and ~~Atom~~ Glen will be ineligible for ~~ta~~ tournament participation.

d

Here we have to run DFS from every node and then we have to check the Highest depth a node can reach.

count = 0

~~for, i in nodes:~~

temp=DFS (~~node~~)

```
def DFS (matrin, node):
    ~~if visited [node] == false:~~
        ~~visited [node] = True~~
    Global count
    for child in matrin [node]:
        if visite [child] == false:
            visited [child] = True
            ~~DFS [matrin, child]~~
            count+ = 1
            DFS [matrin, child]
    return count, node
```

```
max=0
name =''
for i in nodes:

    global count

    count = 0

    temp = DFS(i)

    if count> max:

        max = count
        name = node.
```