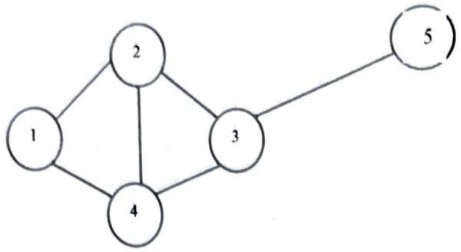
CSC382 --- Final Fall 2023

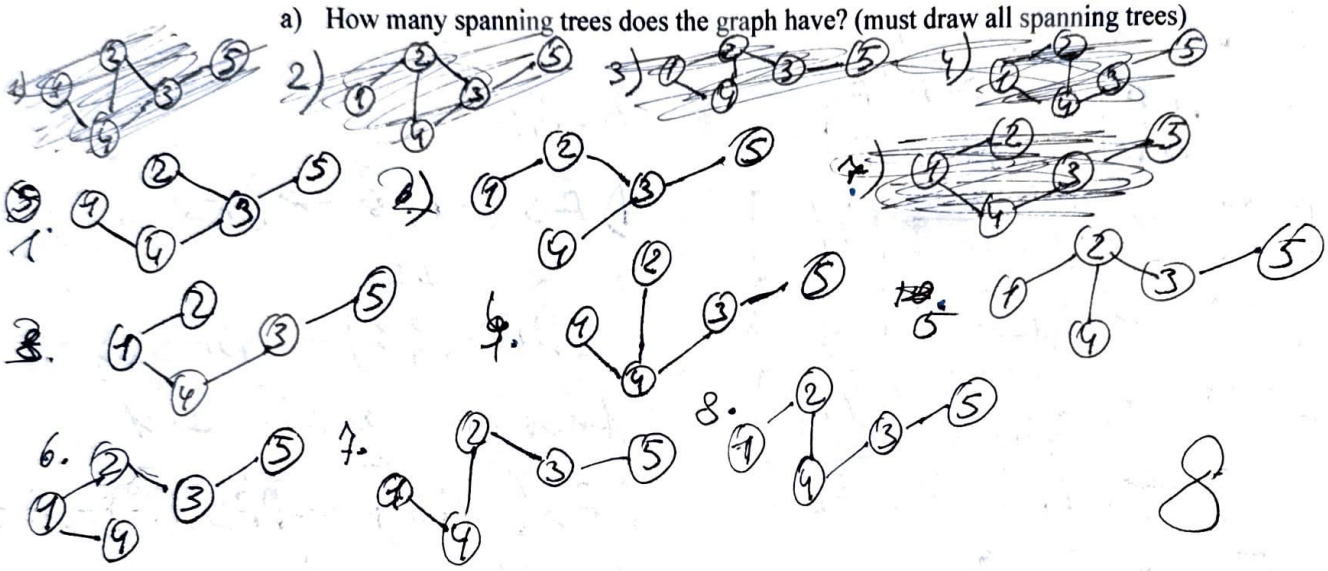
Name: Vitaliy Prymak SS#:

Questions

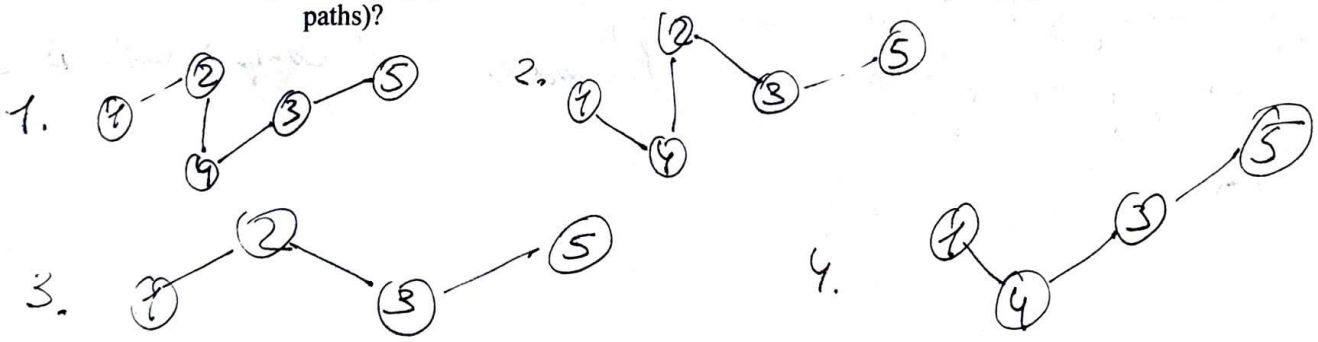
1) Consider the following graph



a) How many spanning trees does the graph have? (must draw all spanning trees)



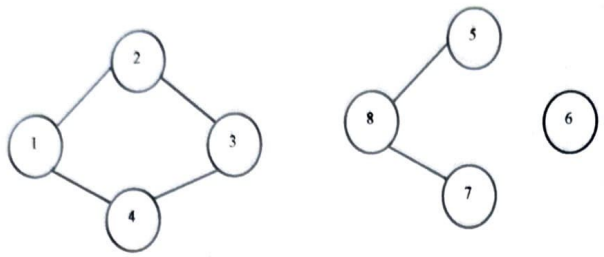
b) How many paths are there between vertex 1 and vertex 5 of the graph (draw the paths)?



CSC382 --- Final Fall 2023

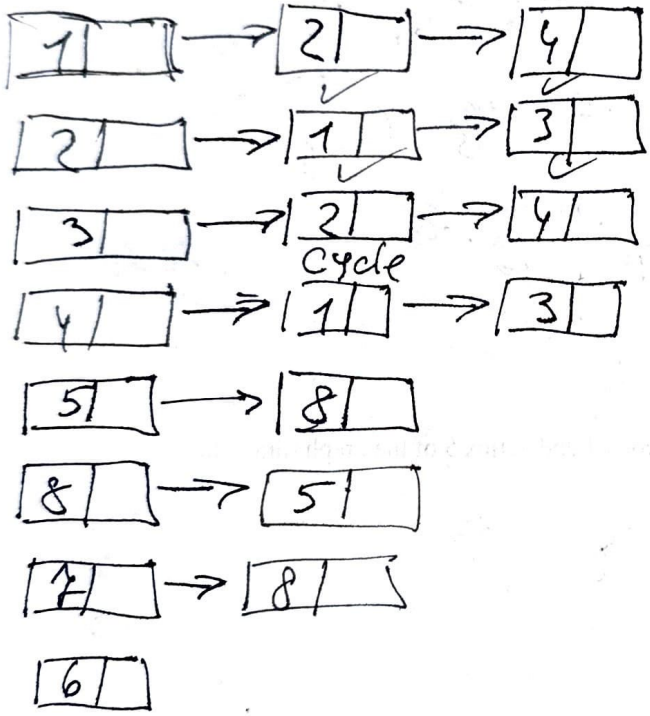
Name : SS#:

2) Using DFS show how to find the number of connected components for the following graph in (your algorithm must show all data-structures: Visited [ ], numofcomponets, etc.). The answer must be clear.




```
numofcomponents = 0
for (int j = 0; j<n; j++){
  if(!visited[j]){
    DFS(j)
    numcomponents++
  }
}
```

Visited	0	1	2	3	4	5	6
	F	F	F	F	F	F	F
	T	T	T	T	T	T	T



main:

DFS(-1; 0) → DFS(1, 2) →

→ DFS(2, 3) → DFS(3, 4)

Another main call

DFS(-1; 5) → DFS(5; 8) → DFS(8, 7)

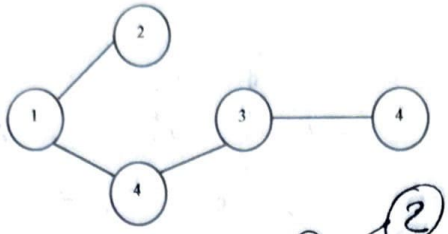
Another main call.

Number of components is 3.

CSC382 --- Final Fall 2023

Name : SS#:

3) Draw all the spanning subgraphs of the following graph:



1. 2. 3. 4. 5. 6. 7. 8. 9. 10 11. 12. 13. 14. 15. 16.

16

CSC382 --- Final Fall 2023

Name : SS#:

4) The following is the pseudocode for DFS to determine if the graph has a cycle or not. Once the global boolean variable **Cycle** is set to **True (originally set to False),** then it remains true. Modify DFS to track the edges of the cycle. Explain your answer (hint: use a queue)

```
void DFS (int v, int p) // p is the parent node
{ Visited [v] = true;
  for (w adjacent to v)
    {
      if (!Visited[w])
          DFS (w,v);
      else {if (w!=p)
              Cycle = True;
    }
  }
```

```
void DFS (int v, int p, queue
<pair<int, int>> & cycleEdges){
visited[v] = true;
for (int w : adj[v]){
  if (!visited[w]){
    parent[w] = v;
    DFS(w, v, cycleEdges); }
else if (w != p && !cycle){
cycle = true;
int (current != w){
cycleEdges.push ({current, parent
[current]});
current = parent[current];}
cycleEdges.push({w, v});
```

```
void addEdge (int u, int v){ // function to
      adj[u].push_back(v);      add edges
      adj[v].push_back(u);
```

```
In main(){
if (cycle){ cout<< "Cycle exists"
while (!cycleEdges.empty()){
  pair<int, int> edge = cycleEdges.front();
  cycleEdges.pop();
cout<< edge.first << " - " < edge.second << endl;
  else { cout << "No cycle found"
```

When cycle is detected, it sets the cycle flag to true and then backtracks through the parent nodes to find edges forming cycle. The edges forming cycle are stored in cycleEdges queue as pairs of nodes (u, v) representing an edge from node u to node v.

CSC382 --- Final Fall 2023

Name : Vitaliy Prymak SS#:

5) Define the following graph-theoretic terms

a) A graph $G=(V,E)$. A graph $G=(V,E)$, is an ordered pair, such that V is finite set whose elements are called vertices and E is a set of unordered pairs of distinct vertices of V, called edges.

b) A spanning subgraph of a graph $G=(V,E)$. A spanning subgraph of a graph $G=(V,E)$, is a subgraph of G, such that $V'=V$ (it contains all original vertices of G)

c) A spanning tree of a graph $G=(V,E)$. A spanning tree of a graph $G=(V,E)$ is a tree $E(V',E')$ that contains all the vertices $G(V'=V)$, but no cycle

d) A trail between vertices u and v of $G=(V,E)$. A trail between vertices u and v of $G=(V,E)$ is a walk without repetition of edges.

e) The degree of a vertex u of a graph $G=(V,E)$. The degree of a vertex v of a graph $G=(V,E)$ is the number of edges incident at v

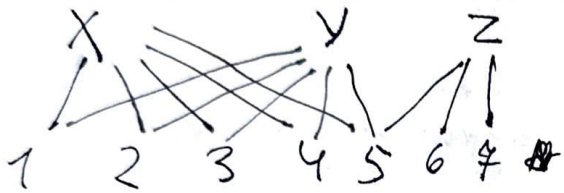
CSC382 --- Final Fall 2023

Name : SS#:

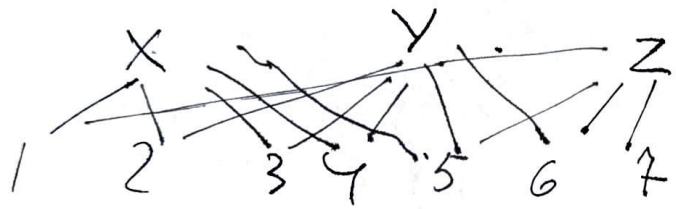
6) A class has 10 students. Suppose that three students X, Y and Z do not know each other. X knows 5 students of the class, Y knows 5 students of the class. At least how many students must Z know to make sure that always there is one student that knows X, Y, and Z? (Explain by using a graph to model the relationships).

Class is 10 students: X, Y, Z and 1, 2, 3, 4, 5, 6, 7;
X knows 5 students and Y knows 5 students

① when X and Y know some students



② when X and Y has 4 common students



③ when X and Y has 3 common students