Topological sort

```
#include<stdio.h>
#define MAX 200
int n,adj[MAX][MAX];
int front = -1,rear = -1,queue[MAX];
void main() {
    int i,j = 0,k;
    int topsort[MAX],indeg[MAX];
    create_graph();
    printf("The adjacency matrix is:\n");
    display();
    for (i=1;i<+n;i++) {
        indeg[i]=indegree(i);
        if(indeg[i]==0)
          insert_queue(i);
    }
    while(front<=rear) {
        k=delete_queue();
        topsort[j++]=k;
        for (i=1;i<=n;i++) {
            if(adj[k][i]==1) {
                adj[k][i]=0;
```

```
                        indeg[i]=indeg[i]-1;
                        if(indeg[i]==0)
                            insert_queue(i);
                    }
                }
        }
        printf("Nodes after topological sorting are:\n");
        for (i=0;i<=n;i++)
          printf("%d",topsort[i]);
        printf("\n");
}
create_graph() {
        int i,max_edges,origin,destin;
        printf("\n Enter number of vertices:");
        scamf("%d",&n);
        max_edges = n * (n - 1);
        for (i = 1;i <= max_edges;i++) {
                printf("\n Enter edge %d (00 to quit):",i);
                scanf("%d%d",&origin,&destin);
                if((origin == 0) && (destin == 0)) {
                        printf("Invalid edge!!\n");
                        i–;
                } else
```

```
            adj[origin][destin] = 1;
    }
    return;
}
display() {
    int i,j;
    for (i = 0;i <= n;i++) {
        for (j = 1;jrear) {
            printf("Queue Underflow");
            return;
        } else {
            del_item = queue[front];
            front = front + 1;
            return del_item;
        }
    }
    int indegree(int node) {
        int i,in_deg = 0;
        for (i = 1;i <= n;i++)
          if(adj[i][node] == 1)
           in_deg++;
        returnin_deg;
    }
```