Topological Sorting

```
#include<stdio.h>
#define SIZE 10 #define MAX
10
int G[SIZE][SIZE], i, j, k; int front, rear; int n, edges;

int b[SIZE], Q[SIZE], indegree[SIZE]; int create() {

front = -1; rear = -1; for (i=0; i<MAX; i++)

//initialising the graph

{ for (j = 0; j<MAX; j++) {


G[i][j] = 0;

} } for (i=0; i<MAX; i++) {

indegree[i] = -99; } n = 5;

edges=7;

G[0][2]=1;

G[0][3]=1;

G[1][0]=1;

G[1][3]=1;

G[2][4]=1;

G[3][2]=1; G[3][4]=1;

return n; } void Display(int

n) { int V1, V2; for (V1= 0;

V1<n; V1++)
{ for (V2= 0; V2<n; V2++)
printf("%d", G[V1][V2]);
printf("\n");
} } void Insert_Q(int vertex, int n)
{ if (rear == n) printf("Queue Overflow\n"); else {
if (front == -1)/*Empty Queue condition*/ front =0;
rear=rear + 1;
Q[rear]=vertex;/* Inserting node into the Q*/
}
} int Delete_Q() { int item; if (front==-
1||front > rear) { printf("Queue
Underflow\n"); return -1; }
else { item=Q[front]; front = front + 1;
return item; } } int Compute_Indeg(int
node, int n)
{ int v1, indeg_count=0; for (v1 =
0; v1<n; v1++) if (G[v1][node]
== 1)//checking for incoming
edge indeg_count++; return
indeg_count++;
} void Topo_ordering(int n)
```

```
{ j = 0; for (i=0; i<n; i++) { indegree[i] =
Compute_Indeg(i, n); if (indegree[i]==0)
Insert_Q(i, n); } while (front
<= rear) { k = Delete_Q();
b[j++] = k; for (i=0; i<n; i++)
{ if (G[k][i]==1)
{ G[k][i] = 0;
indegree[i]=indegree[i] - 1; if
(indegree[i]==0)
Insert_Q(i, n);
}
} } printf("\nThe result of after topological sorting is ..."); for
(i=0; i<n; i++) printf("%d",b[i]); printf("\n");
} int main() { n = create(); printf("The
adjacency matrix is : \n");
Display(n); Topo_ordering(n);
return 0;}
```

Output : The adjacency matrix

```
is:00110 10010 00001 00101 00000

The result of after topological

sorting is ...10324
```