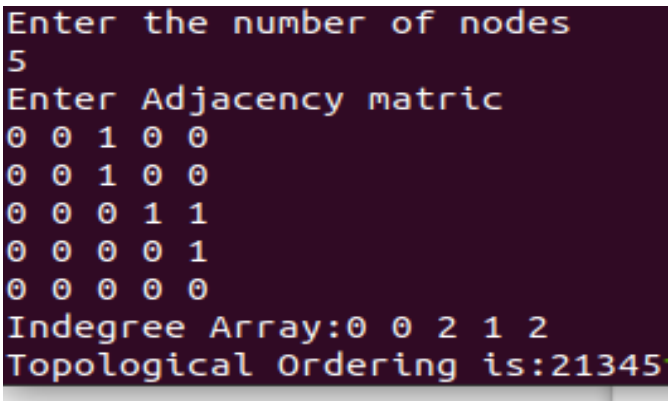
Design and implement C/C++ Program to obtain the Topological ordering of vertices in a given digraph.

```
#include<stdio.h>
void ts(int a[20][20], int n)
{
	int t[10],vis[10],stack[10],i,j,indeg[10],top=0,ele,k=1;
	for(i=1;i<=n;i++)
	{
		t[i]=0;
		vis[i]=0;
		indeg[i]=0;
	}
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			if(a[i][j]==1)
			{
				indeg[j]=indeg[j]+1;
			}
		}
	}
	printf("Indegree Array:");
	for(i=1;i<=n;i++)
	printf("%d ",indeg[i]);
	for(i=1;i<=n;i++)
	{
		if(indeg[i]==0)
		{
			stack[++top]=i;
			vis[i]=1;
		}
	}
	while(top>0)
	{
		ele=stack[top--];
		t[k++]=ele;
		for(j=1;j<=n;j++)
		{
			if(a[ele][j]==1 && vis[j]==0)
			{
				indeg[j]=indeg[j]-1;
				if(indeg[j]==0)
```

```
                        {
                                stack[++top]=j;
                                vis[j]=1;
                        }
                }
        }
        printf("\nTopological Ordering is:");
        for(i=1;i<=n;i++)
        printf("%d",t[i]);
}

int main()
{
        int n,a[20][20],i,j;
        printf("Enter the number of nodes\n");
        scanf("%d",&n);
        printf("Enter Adjacency matric\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        scanf("%d",&a[i][j]);
                }
        }
        ts(a,n);
}
```

OUTPUT:



Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic Programming method.

Knapsack()

```
// Input : n-number of items,W-capacity of the knapsack, v-profits – all integers
// Output :V(n,W)
// Initialization of first column and first row elements

        repeat for
        i=0 to nset
        V(i,0)=0
        repeat for
        j=0 to Wset
        V(0,j)=0
//complete remaining entries row by row repeat for i=1 to
        nrepeat for j=1 to W
        if (wi<=j)
        V(i,j)=max{V(i-1,j),V(i-1,j-wi)+vi if
        (wi>j)V(i,j)=V(i-1,j)
        printV(n,W)
```

Program:

```
#include<stdio.h>
int w[10],p[10],n;

int max(int a,int b)
{
        return a>b?a:b;
}
int knap(int i,int m)
{
        if(i==n) return w[i]>m?0:p[i];
        if(w[i]>m) return knap(i+1,m);
        return max(knap(i+1,m),knap(i+1,m-w[i])+p[i]);
}
int main()
{
        int m,i,max_profit;
        printf("\nEnter the no. of objects:");
        scanf("%d",&n);
        printf("\nEnter the knapsack capacity:");
        scanf("%d",&m);
        printf("\nEnter profit followed by weight:\n");
        for(i=1;i<=n;i++)
        scanf("%d %d",&p[i],&w[i]);
        max_profit=knap(1,m);
        printf("\nMax profit=%d",max_profit);
        return 0;
}
```

OUTPUT: