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

```
#include<stdio.h>
int temp[10],k=0;
void sort(int a[][10],int id[],int n)
{
        int i,j;
        for(i=1; i<=n; i++)
        {
                if(id[i]==0)
                {
                        id[i]=-1;
                        temp[++k]=i;
                        for(j=1; j<=n; j++)
                        {
                                if(a[i][j]==1 && id[j]!=-1)
                                id[j]--;
                        }
                        i=0;
                }
        }
}
void main()
{
        int a[10][10],id[10],n,i,j;
        printf("\nEnter the n value:");
        scanf("%d",&n);
        for(i=1; i<=n; i++)
        id[i]=0;
        printf("\nEnter the graph data:\n");
        for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
        {
                scanf("%d",&a[i][j]);
                if(a[i][j]==1)
                id[j]++;
        }
        sort(a,id,n);
        if(k!=n)
        printf("\nTopological ordering not possible");
        else
        {
                printf("\nTopological ordering is:");
                for(i=1; i<=k; i++)
                printf("%d ",temp[i]);
        }
}
```

11.Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>

int merge(int b[],int c[],int a[],int p,int q,int n)

{

        int i,j,k;

        i=j=k=0;

        while(i<p && j<q)

        {

                if(b[i]<=c[j])

                {

                        a[k]=b[i];

                        i++;

                }

                else

                {

                        a[k]=c[j];

                        j++;

                }

                k++;

        }

        if(i==p)

        {

                while(j<q)
                {
                        a[k]=c[j];
                        k++;
                        j++;
                }
        }
        else
        {
                while(i<p && k<n)
                a[k++]=b[i++];
        }
}
int mergesort(int a[],int n)
{
        int b[n/2];
        int c[n-n/2];
        int i,j;
        if(n>1)
        {
                for(i=0;i<n/2;i++)
                b[i]=a[i];
                for(i=n/2,j=0;i<n;i++,j++)
                c[j]=a[i];
                mergesort(b,n/2);
                mergesort(c,n-n/2);
                merge(b,c,a,n/2,n-n/2,n);
        }
}
```

```
int main()
{
	int temp,min,j,i,n,a[100000],choice;
	clock_t t;
	printf("enter the number of elements :");
	scanf("%d",&n);
	printf("1. Read from file 2. Random numbers");
	scanf("%d",&choice);
	switch(choice)
	{
		case 1: printf("file numbers\n");
			FILE *file = fopen("num.txt","r");
			int i=0;
			while(!feof(file) && i<n)
			{
				//printf("%d ",i+1);
				fscanf(file,"%d",&a[i]);
				printf("%d\n",a[i]);
				i++;
			}
			fclose(file);
			break;
		case 2: printf("Random number generator");
			for(i=0;i<n;i++)
			{
				a[i] = rand()%1000;
				printf("%d\n",a[i]);
			}
```

```
                break;
        }
        t = clock();
        mergesort(a,n);
        t = clock()-t;
        double time =((double)t)/CLOCKS_PER_SEC;
        printf("entered number after sorting\n");
        for (i=0;i<n;i++)
        printf("%d\n",a[i]);
        printf("sort function took %f sec to execute",time);
        return 0;
}
```

12. Design and implement C/C++ Program for N Queen's problem using Backtracking.

```
#include <stdio.h>
#include <stdlib.h>
int x[10];
int place(int k,int i)
{
        int j;
        for(j=1;j<=k-1;j++)
        if(x[j]==i || abs(x[j]-i)==abs(j-k))
        return 0;
        return 1;
}
void display(int n)
{
        int k,i,j;
        char cb[n][n];
        for(k=1;k<=n;k++)
        cb[k][x[k]]='Q';
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        if(j!=x[i])
                        cb[i][j]='-';
                }
        }
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                printf("%c\t",cb[i][j]);
                printf("\n");
        }
        printf("\n\n");
}




void NQueens(int k,int n)
{
        int i;
        for(i=1;i<=n;i++)
        if(place(k,i))
        {
                x[k]=i;
                if(k==n)
                {
                        printf("Solution\n");
                        display(n);
                }
                else
                NQueens(k+1,n);
        }
}
int main(void)
```

```
{
        int n,k=1;
        printf("Enter the dimensions of the chessboard\n");
        scanf("%d",&n);
        if(n==2 || n==3)
        {
                printf("No solution\n");
                exit(0);
        }
        NQueens(k,n);
        return 0;
}
```