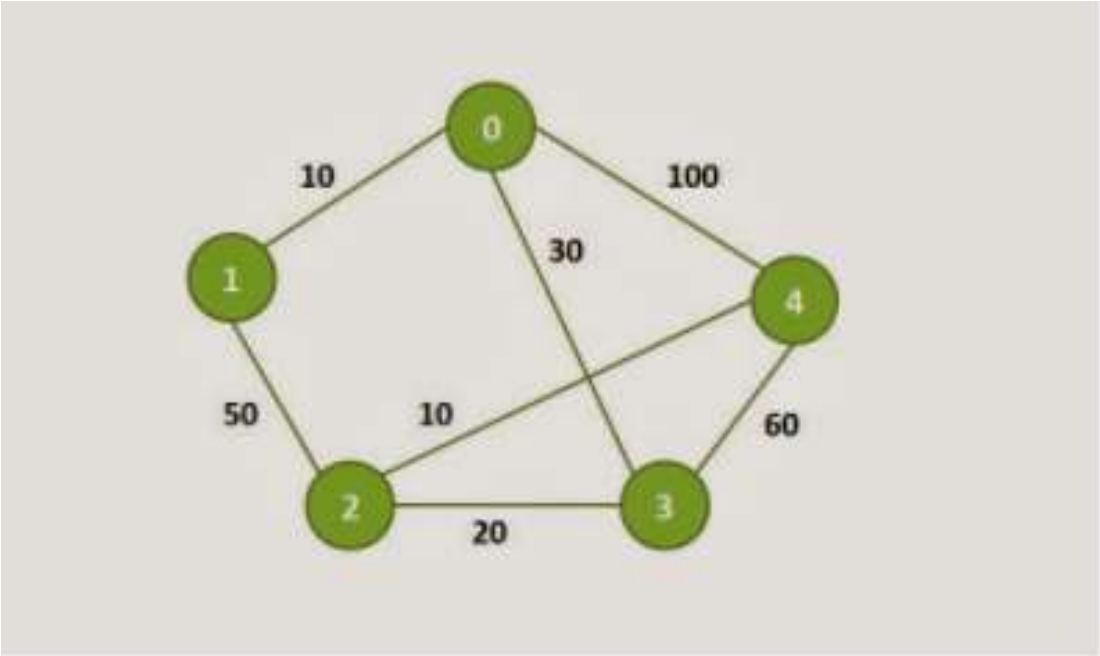
Programme	:	B.Tech (CSE)	Semester	:	Fall 2020-2021
Course	:	Data Structures and Algorithm	Code	:	CSE2003
Faculty	:	Prof Sandeep	Slot	:	L67+68
Name	:	Wilson Vidyut Doloy	Reg No	:	19BCE1603

31/10/2020

DSA Graph Lab 3

Q-1) Find the single source shortest path of the below graph using Dijkstra's algorithm.



CODE:

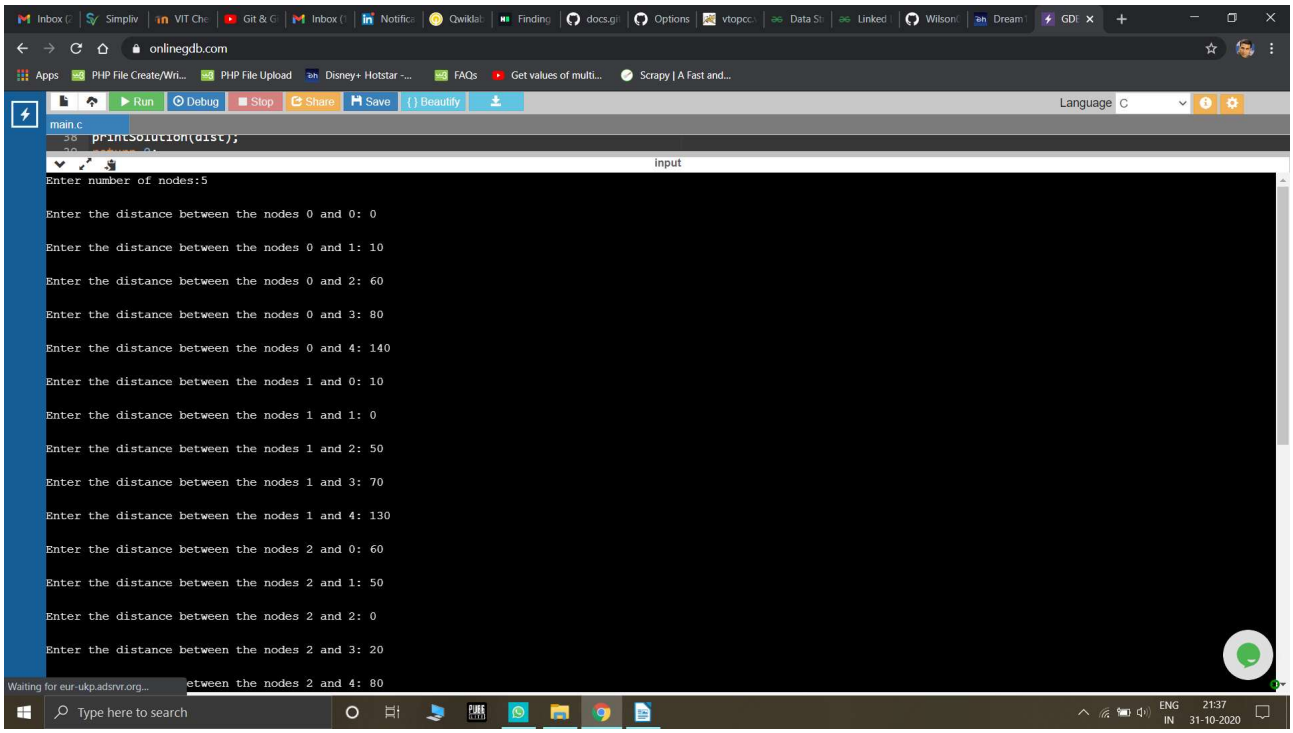
```
#include<stdio.h>

#include<stdbool.h>

#include<limits.h>
```

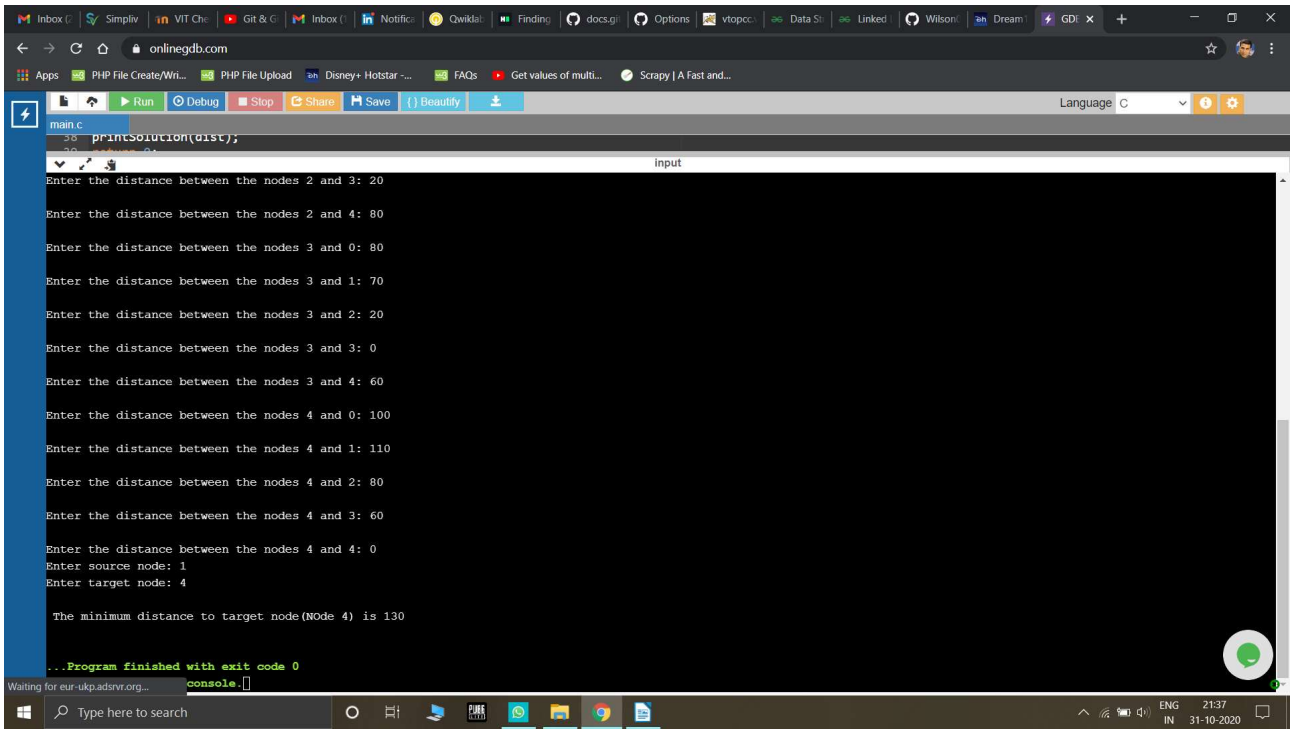
```
#define V 100
int num;
int network[100][100];
int target , source;
int dist[V];
int minDistance(int dist[], bool sptSet[])
{
        int min = INT_MAX, min_index;
        for(int v=0;v<num; v++)
        if(sptSet[v]==false && dist[v]<=min)
        min=dist[v],min_index=v;
        return min_index;
}
int printSolution(int dist[])
{
        printf("\n The minimum distance to target node(NOde %d) is %d\n",
        target, dist[target]);
        return 0;
}
int dijkstra(int network[V][V], int src)
{
bool sptSet[V];
for(int i=0;i<V;i++)
        dist[i]=INT_MAX, sptSet[i]=false;
dist[src]=0;
for(int count=0;count<num-1;count++)
{
        int u=minDistance(dist,sptSet);
```

```
            sptSet[u]=true;
            for(int v=0;v<num;v++)
            if(!sptSet[v] && network[u][v] && dist[u]!=INT_MAX &&
            dist[u]+network[u][v]<dist[v])
            dist[v]=dist[u]+network[u][v];
}
printSolution(dist);
return 0;
}

int main()
{

            printf("Enter number of nodes:");
            scanf("%d",&num);

            //printf("Enter adj matrix:\n");
            for(int i=0;i<num;i++)
            {
                        for(int j=0; j<num; j++)
                        {
                          printf("\nEnter the distance between the nodes %d and %d: ", i, j);
          scanf("%d",&network[i][j]);
                        }
            }

            printf("Enter source node: ");
            scanf("%d",&source);
```

```
        printf("Enter target node: ");
        scanf("%d",&target);
    dijkstra(network,source);
}
```

OUTPUT:


onlinegdb.com
Run
Debug
Stop
Share
Save
{} Beautify
Language C
main.c
printSolution(dist);
input
Enter number of nodes:5
Enter the distance between the nodes 0 and 0: 0
Enter the distance between the nodes 0 and 1: 10
Enter the distance between the nodes 0 and 2: 60
Enter the distance between the nodes 0 and 3: 80
Enter the distance between the nodes 0 and 4: 140
Enter the distance between the nodes 1 and 0: 10
Enter the distance between the nodes 1 and 1: 0
Enter the distance between the nodes 1 and 2: 50
Enter the distance between the nodes 1 and 3: 70
Enter the distance between the nodes 1 and 4: 130
Enter the distance between the nodes 2 and 0: 60
Enter the distance between the nodes 2 and 1: 50
Enter the distance between the nodes 2 and 2: 0
Enter the distance between the nodes 2 and 3: 20
Waiting for eur-ukp.adsrvr.org...
Type here to search
ENG
IN
21:37
31-10-2020


onlinegdb.com
Run
Debug
Stop
Share
Save
{} Beautify
Language C
main.c
printSolution(dist);
input
Enter the distance between the nodes 2 and 3: 20
Enter the distance between the nodes 2 and 4: 80
Enter the distance between the nodes 3 and 0: 80
Enter the distance between the nodes 3 and 1: 70
Enter the distance between the nodes 3 and 2: 20
Enter the distance between the nodes 3 and 3: 0
Enter the distance between the nodes 3 and 4: 60
Enter the distance between the nodes 4 and 0: 100
Enter the distance between the nodes 4 and 1: 110
Enter the distance between the nodes 4 and 2: 80
Enter the distance between the nodes 4 and 3: 60
Enter the distance between the nodes 4 and 4: 0
Enter source node: 1
Enter target node: 4
The minimum distance to target node(NOde 4) is 130
...Program finished with exit code 0
Press ENTER to exit console.
Waiting for eur-ukp.adsrvr.org...
Type here to search
ENG
IN
21:37
31-10-2020

Q-2) Find the all pair shortest path using Floyd Warshall algorithm using the above graph.

CODE:

```
#include<stdio.h>

int min(int,int);

void floyds(int p[10][10],int n)

{

int i,j,k;

for(k=1;k<=n;k++)

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(i==j)

p[i][j]=0;

else

p[i][j]=min(p[i][j],p[i][k]+p[k][j]);

}
```

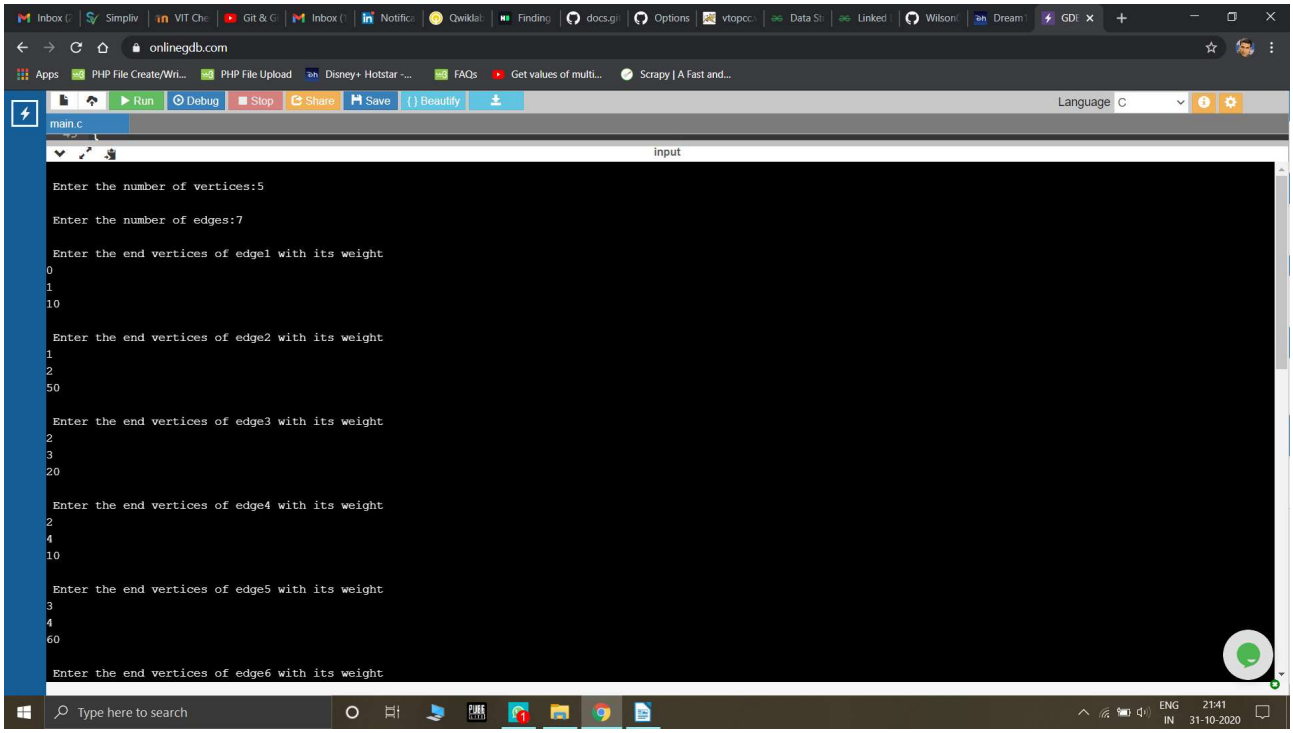
```
int min(int a,int b)

{

if(a<b)

return(a);

else

return(b);

}

int main()

{

int p[10][10],w,n,e,u,v,i,j;;

printf("\n Enter the number of vertices:");

scanf("%d",&n);

printf("\n Enter the number of edges:\n");

scanf("%d",&e);

for(i=1;i<=n;i++)
```

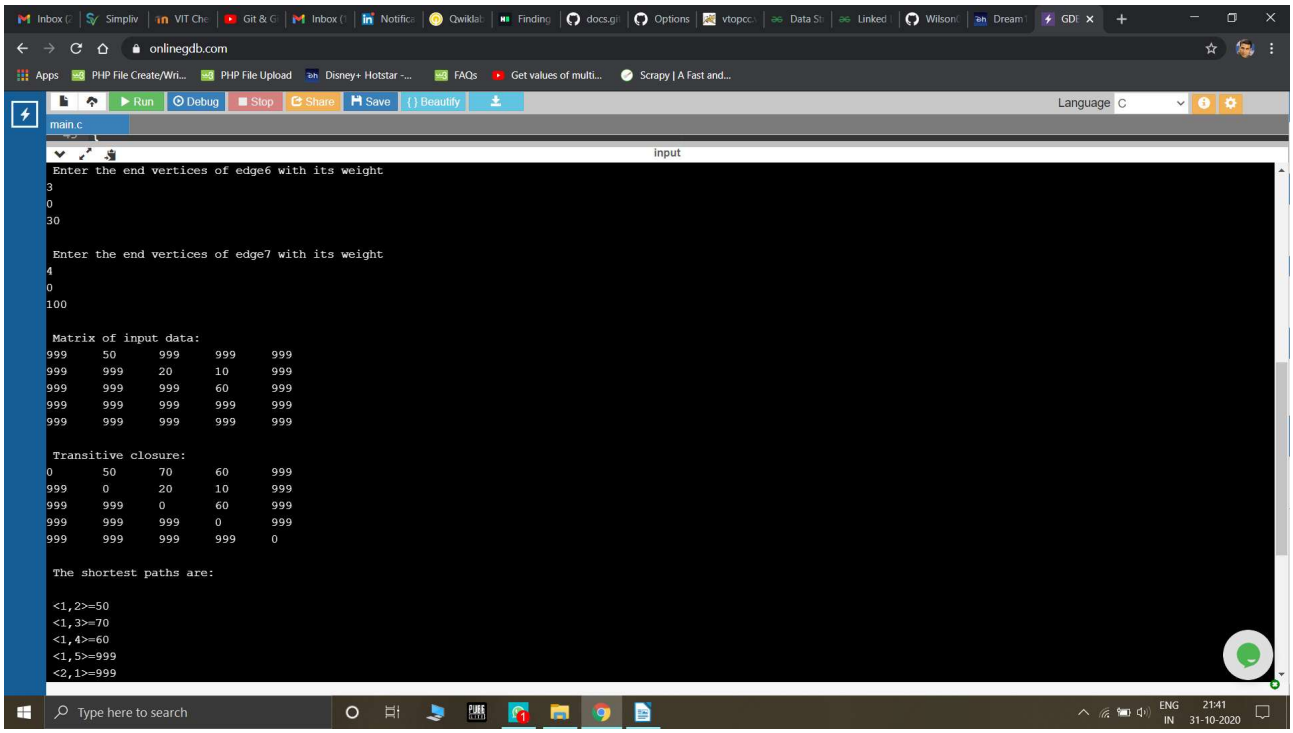
```
{
for(j=1;j<=n;j++)
p[i][j]=999;
}
for(i=1;i<=e;i++)
{
printf("\n Enter the end vertices of edge%d with its weight \n",i);
scanf("%d%d%d",&u,&v,&w);
p[u][v]=w;
}
printf("\n Matrix of input data:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
printf("%d \t",p[i][j]);
```

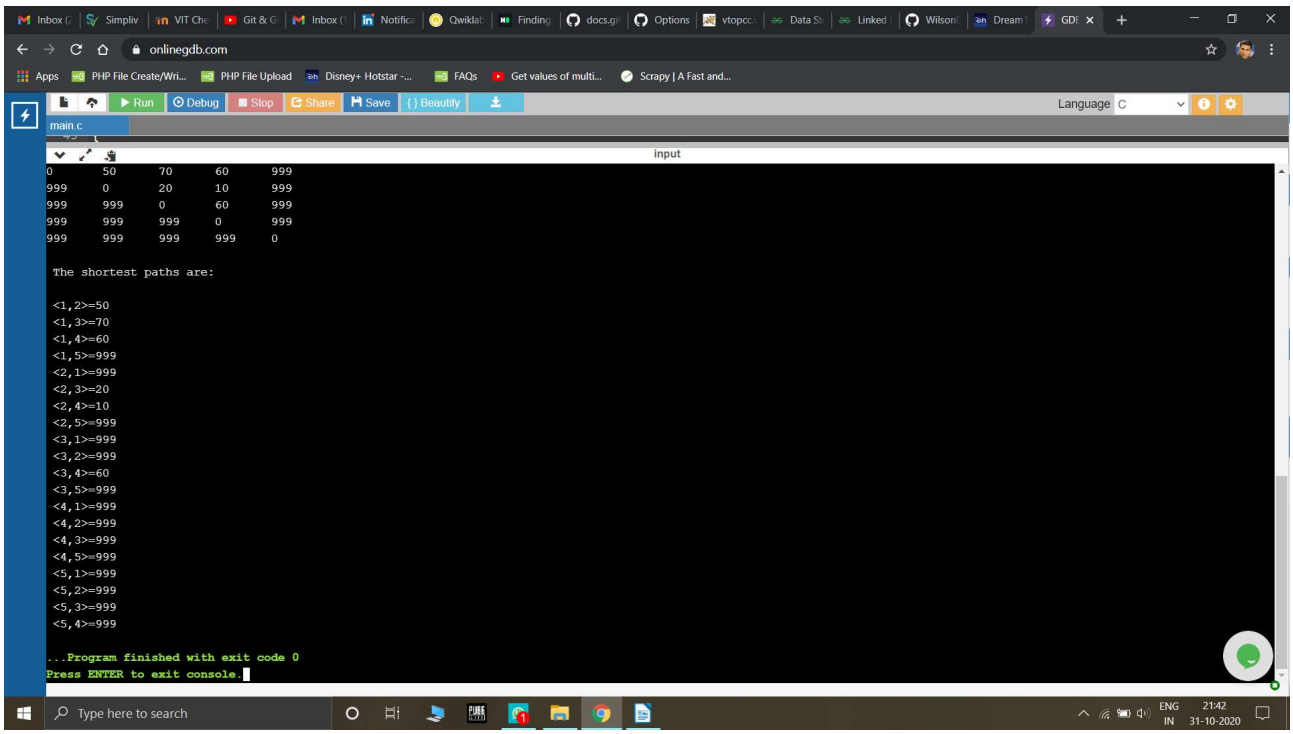
```
printf("\n");

}

floyds(p,n);

printf("\n Transitive closure:\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

printf("%d \t",p[i][j]);

printf("\n");

}

printf("\n The shortest paths are:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

{

if(i!=j)
```

```
printf("\n <%d,%d>=%d",i,j,p[i][j]);

}

}
```

OUTPUT:




onlinegdb.com
Run
Debug
Stop
Share
Save
{} Beautify
Language C
main.c
input
Enter the end vertices of edge6 with its weight
3
0
30
Enter the end vertices of edge7 with its weight
4
0
100
Matrix of input data:
999 50 999 999 999
999 999 20 10 999
999 999 999 60 999
999 999 999 999 999
999 999 999 999 999
Transitive closure:
0 50 70 60 999
999 0 20 10 999
999 999 0 60 999
999 999 999 0 999
999 999 999 999 0
The shortest paths are:
<1,2>=50
<1,3>=70
<1,4>=60
<1,5>=999
<2,1>=999
Type here to search
ENG
IN
21:41
31-10-2020


onlinegdb.com
Run
Debug
Stop
Share
Save
{} Beautify
Language C
main.c
input
0 50 70 60 999
999 0 20 10 999
999 999 0 60 999
999 999 999 0 999
999 999 999 999 0
The shortest paths are:
<1,2>=50
<1,3>=70
<1,4>=60
<1,5>=999
<2,1>=999
<2,3>=20
<2,4>=10
<2,5>=999
<3,1>=999
<3,2>=999
<3,4>=60
<3,5>=999
<4,1>=999
<4,2>=999
<4,3>=999
<4,5>=999
<5,1>=999
<5,2>=999
<5,3>=999
<5,4>=999
...Program finished with exit code 0
Press ENTER to exit console.
Type here to search
ENG
IN
21:42
31-10-2020