Name	Harsh Chandra
UID no.	2021700013
Experiment No.	9

AIM:	**Approximation algorithms (Travelling Salesman Problem)**
Program 1	
ALGORITHM/ THEORY:	• Travelling salesman problem takes a graph G {V, E} as an input and declare another graph as the output (say G') which will record the path the salesman is going to take from one node to another. • The algorithm begins by sorting all the edges in the input graph G from the least distance to the largest distance. • The first edge selected is the edge with least distance, and one of the two vertices (say A and B) being the origin node (say A). • Then among the adjacent edges of the node other than the origin node (B), find the least cost edge and add it onto the output graph. • Continue the process with further nodes making sure there are no cycles in the output graph and the path reaches back to the origin node A. • However, if the origin is mentioned in the given problem, then the solution must always start from that node only. Let us look at some example problems to understand this better.

PROGRAM:	#include <stdio.h>
	int matrix[25][25], visited_cities[10], limit, cost = 0;
	
	int tsp(int c)
	{
	int count, nearest_city = 999;
	int minimum = 999, temp;
	for(count = 0; count < limit; count++)
	{
	if((matrix[c][count] != 0) && (visited_cities[count] == 0))
	{

```
 if(matrix[c][count] < minimum)
 {
 minimum = matrix[count][0] + matrix[c][count];
 }
 temp = matrix[c][count];
 nearest_city = count;
 }
 }
 if(minimum != 999)
 {
 cost = cost + temp;
 }
 return nearest_city;
}

void minimum_cost(int city)
{
 int nearest_city;
 visited_cities[city] = 1;
 printf("%d ", city + 1);
 nearest_city = tsp(city);
 if(nearest_city == 999)
 {
 nearest_city = 0;
 printf("%d", nearest_city + 1);
 cost = cost + matrix[city][nearest_city];
 return;
 }
 minimum_cost(nearest_city);
}

int main()
{
 int i, j;
 printf("Enter Total Number of Cities:\t");
 scanf("%d", &limit);
 printf("\nEnter Cost Matrix\n");
 for(i = 0; i < limit; i++)
 {
 printf("\nEnter %d Elements in Row[%d]\n", limit, i + 1);
```

```
for(j = 0; j < limit; j++)
{
scanf("%d", &matrix[i][j]);
}
visited_cities[i] = 0;
}
printf("\nEntered Cost Matrix\n");
for(i = 0; i < limit; i++)
{
printf("\n");
for(j = 0; j < limit; j++)
{
printf("%d ", matrix[i][j]);
}
}
printf("\n\nPath:\t");
minimum_cost(0);
printf("\n\nMinimum Cost: \t");
printf("%d\n", cost);
return 0;
}
```

RESULT:

```
Enter Total Number of Cities:   4

Enter Cost Matrix

Enter 4 Elements in Row[1]
1 2 3 4

Enter 4 Elements in Row[2]
5 6 7 8

Enter 4 Elements in Row[3]
3 4 5 6

Enter 4 Elements in Row[4]
9 8 4 3

Entered Cost Matrix

1 2 3 4
5 6 7 8
3 4 5 6
9 8 4 3

Path:   1 4 3 2 1

Minimum Cost:   17


...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION:	From this experiment, I understood the implementation and solution of travelling salesman problem using approximate algorithms.