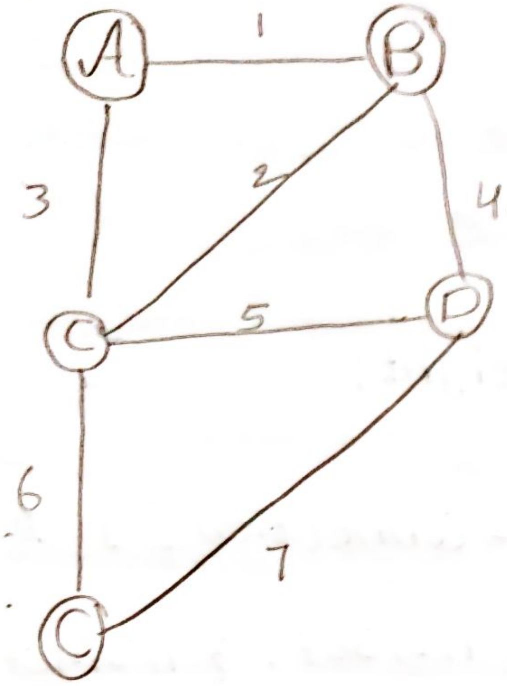
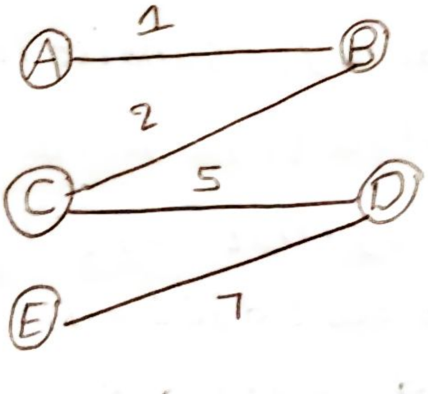
lab Experiment - 7

R. Nanda Kishore Reddy
2320030130

Pre lab:-



Prims algorithm:-



1+2+5+7 = 15

$\textcircled{A}$

$A \to \textcircled{B},\ C$

1 3

$A \to B$

$B \to \textcircled{C}\ D$

2 , 4

$B \to C$

$C \to \overset{\times}{B}, \textcircled{D}$

2 5

$C \to \textcircled{D}$

$D \to \textcircled{E}$

[illegible]

EXP-7

R.Nandakishore
Reddy
2320030130

In Lab:-

```
1.# include<stdio.h>
  # include <string.h>
  struct Edge {
    int src, dest, weight;
  };
  struct Edge cable-Connection [ ] = {
 { 0,1,10}, { 0,2,20}. { 1,2,30} . { 1,3,40}
 {2,3,50}.{3,4,60}.{2,4,70}
 };
char* apartments [ ] = { "Aparna Amaravati",
 "jayabheri", "vajra residency", "sunrise Tower",
 "dandamudi enclave", "Aditya villa grande"}
 int parent[5], rank[5] mst[5],[3],
 mst_size=0, total_cost =0, edges = 7,
apartments_ Count=5; void Sortedges(){
for (int i=0; i< edges-1 ; i++){
for (int j=0; j<edges -i-1; j++){
if (cable_Connections[j] weight > Cable-
Connections[j+1] .weight){
Struct Edge temp = Cable Connections[j];
Cable - Connections [j] = Cable Connections [j+1];
Cable- Connections [j+1]=temp;
 } } } }
```

R.Nanda Kishore
Reddy
2320030130

```
int find (int u){
if (u!=parent[u])
  parent[u]=find (parent[u]);
 return parent[u];
 }
Void union_sets(int u, int v){
if (rank[u] > rank[v])
   parent[v]=u;
 else if (rank(u) < rank[v])
  parent[u]=v;
 else{
parent[v]=u;
rank[u]++;
} }
int main(){
for(int i=0; i<apartments_count; i++){
   parent[i]=i;
 rank[i]=0;
 }
 sort-edges();
for (int i=0; i<edges; i++){
int u = find (cable_connections[i].src);
int v = find (cable-connections[i] .dest);
if (u!=v){
mst[mst-size][0]=cable-connections[i]-src;
mst[mst-size][1] =cable-connections [i] dest;
mst[mst-size][2]=cable-connections[i].weight;
total cost += cable-connections [i].weight;
mst-size++;
union-sets(u,v);
  }
}
Printf ("Total Cost: %d\n MST"; [Total-Cost]);
for (int i=0;i<mst-size;i++){
Printf [("%s, %s, %d"; apartments [dest [i][0]];
apartments[mst[i][1]], mst [i][2];
 if (i<mst_size -1)
  Printf (", ");
 }
 printf (" ]/n");
 return 0;
 }
```

R.Nanda
Kishore Redd
2320030131

```
2.#include<stdio.h>
 #include<stdlib.h>
 #define V5

int minDistance (intalist[], int sptset[]){
 int mine=INT-MAX, min_index;
 for(int V=0; V<V; V++)
 if (sptset[V]=0 && dist[V] <=min)
  min=dist[V], min_index=V;
 return min-index;
 }
 Void dijkstra(int graph[V][V], int src){
 int dist[V], sptset[V];
 for (int i=0; i<V ; i++){
```

R.Nanda
kishore Reddy
2320030130

```
dist[i] =INT_MAX, sptset[i]=0;
dist[src]=0;
for (int count=0; count<V-1; count++){
 int u= minDistance(dist, sptset);
 sptset[u]=1
 for (!sptset[v] && graph[u][v] && dist[u]!=
 INT_MAX && dist[u] + graph [u][v] < dist[v])
  dist [v] = dist[u] +graph [u] [v]
  }
 printf ("shortest paths from w:\n");
 for (int i=0; i<n; i++)
 printf ("w -> (%d :%d\n ", i, dist[i]);
}
int main ( ) {
int graph[V] [V]={{0,3,6,0,0}, {3,0,0,2,0}
{6,0,0,4,2}, {0,2,4,0,1}, {0,0,2,1,0} }
 dijkstra(graph, 0);
 return 0;
 }
```

OUTPUT :-

```
shortest Paths from w:

w -> C0:0
w -> C1 : 3
w -> C2 : 6
w -> C3 : 5
w -> C4 : 6
```

[illegible]