Exp [illegible] implementation of Minimum spanning Tree using kruskal Algorithm.

Aim: To implementation of Minimum spanning Tree using kruskal Algorithm.

Algorithm:

* . start
* . input no.of vertices and edges.
* . List all edges and sort by weight.
* . ~~List~~ use union-find to detect cycles.
* . Add edges to MST if no cycles is formed.
* . Repeat Until MST is formed.
* . print MST and cost.
* . stop.

program:

```
#include <stdio.h>
#define V 5
#define E 7
 struct edge { int u, v, w; } e[E];
 int parent [V];
 int find(int x) {
     while (parent [x] != x) x = parent [x];
     return x;
 }
 void uni (int x, int y) {
     parent [find (x)] = find(y);
 }
   int main() {
         struct edge temp;
          int i,j;
      struct edge list [E] = {
          {0, 1, 2}, {1, 2, 3}, {0, 3, 6},
          {1, 4, 5}, {2, 4, 7}, {3, 4, 9}, {2, 3, 2}
      };
      for (i= 0; i< E; i++) e[i] = list [i];
      for (i=0; i<V; i++) parent [i] = i;
```

S.Viv[redacted]anth Reddy

```
for (i=0; i<E; i++)
    for (j=0; j<E-i-1; j++)
        if (e[j].w > e[j+1].w){
            temp=e[j]; e[j] = e[j+1]; e[j+1] = temp;
    }
  printf("MST edges:\n");
  int count =0;
  for (i=0; i<E && && count <V-1; i++){
      int u = e[i].u, v = e[i].v;
      if ( find (u) ! = find (v)){
         printf ("%d-%d (w=%d)\n", u, v, e[i].w);
         uni (u, v);
         count ++;
    }
  }
}
```

<u>**output:**</u>

```
MST edges:
 0 - 1 (w=2)
 2 - 3 (w=2)
 1 - 2 (w=3
 1 - u (w=5)
```

<u>**Result.**</u> Thus, the program executed successfully