//Minimum Spanning Tree

//Prim's Algorithm

//Complexity : O(E logV)

#define MAX 100

vector<int> G[MAX], W[MAX];

priority\_queue<pair<int, int> >pq;

bitset<MAX>taken;

void process(int u) {

//mark this node as taken

taken[u] = 1;

//get all the edges of this node on a priority queue

for(int i = 0; i < G[u].size(); i++) {

int v = G[u][i];

int w = W[u][i];

if(!taken[v])

pq.push(make\_pair(-w, -v));

}

//priority queue returns the minimum node first, if tie, then the first node

}

int main() {

int V, E, u, v, w;

scanf("%d %d", &V, &E);

while(E--) {

scanf("%d %d %d", &u, &v, &w);

G[u].push\_back(v);

W[u].push\_back(w);

G[v].push\_back(u);

W[v].push\_back(w);

}

//Main Prim's MST code

taken.reset();

process(0); //taking 0 node as default

int mst\_cost = 0;

while(!pq.empty()) {

w = -pq.top().first;

v = -pq.top().second;

pq.pop();

if(!taken[v]) { //if the node is not taken, then use this node

mst\_cost += w; //as it contains the minimum edge

process(v);

}

}

printf("Prim's MST cost : %d\n", mst\_cost);

return 0;

}