Prim in C++

```
#include <bits/stdc++.h>
using namespace std;

class Solution
{
public:
	//Function to find sum of weights of edges of the Minimum Spanning Tree.
	int spanningTree(int V, vector<vector<int>> adj[])
	{
		priority_queue<pair<int, int>,
			vector<pair<int, int> >, greater<pair<int, int>>> pq;

		vector<int> vis(V, 0);
		// {wt, node}
		pq.push({0, 0});
		int sum = 0;
		while (!pq.empty()) {
			auto it = pq.top();
			pq.pop();
			int node = it.second;
			int wt = it.first;

			if (vis[node] == 1) continue;
			// add it to the mst
			vis[node] = 1;
			sum += wt;
			for (auto it : adj[node]) {
				int adjNode = it[0];
				int edW = it[1];
				if (!vis[adjNode]) {
					pq.push({edW, adjNode});
				}
			}
		}
		return sum;
	}
};


int main() {

	int V = 5;
	vector<vector<int>> edges = {{0, 1, 2}, {0, 2, 1}, {1, 2, 1}, {2, 3, 2}, {3, 4, 1}, {4, 2, 2}};
	vector<vector<int>> adj[V];
	for (auto it : edges) {
		vector<int> tmp(2);
		tmp[0] = it[1];
		tmp[1] = it[2];
		adj[it[0]].push_back(tmp);

		tmp[0] = it[0];
		tmp[1] = it[2];
		adj[it[1]].push_back(tmp);
	}

	Solution obj;
```

Input Edges

```
edges = {
  {0, 1, 2},
  {0, 2, 1},
  {1, 2, 1},
  {2, 3, 2},
  {3, 4, 1},
  {4, 2, 2}
}
```

🔁 Adjacency List

Node	Neighbors
0	[1,2], [2,1]
1	[0,2], [2,1]
2	[0,1], [1,1], [3,2], [4,2]
3	[2,2], [4,1]
4	[3,1], [2,2]

🧠 Prim's MST Logic (Min-Heap)

We track:

- pq: min-heap for {weight, node}
- vis[]: visited array
- sum: total MST weight

📊 Dry Run Table

Step	pq (Min-Heap)	node	wt	vis	sum	Action Taken
1	{(0, 0)}	0	0	[1, 0, 0, 0, 0]	0	Add node 0, add neighbors 1 (wt=2), 2 (wt=1) to pq
2	{(1, 2), (2, 1)}	2	1	[1, 0, 1, 0, 0]	1	Add node 2, add unvisited neighbors: 1(wt=1), 3(wt=2), 4(wt=2)
3	{(1, 1), (2, 1), (2, 3), (2, 4)}	1	1	[1, 1, 1, 0, 0]	2	Add node 1, skip already visited 0 & 2
4	{(2, 1), (2, 3), (2, 4)}	1	2	Already visited	-	Skip

```
        int sum = obj.spanningTree(V, adj);
        cout << "The sum of all the edge weights: " <<
sum << endl;

        return 0;
}
```

Step	pq (Min-Heap)	node	wt	vis	sum	Action Taken
5	{(2, 3), (2, 4)}	3	2	[1, 1, 1, 1, 0]	4	Add node 3, add neighbor 4 (wt=1)
6	{(1, 4), (2, 4)}	4	1	[1, 1, 1, 1, 1]	5	Add node 4, skip visited 3, 2
7	{(2, 4)}	4	2	Already visited	-	Skip

✅ **Final Result:**

Variable	Value
sum	**5**
vis	[1,1,1,1,1] (All visited)

✅ **Output:**

The sum of all the edge weights: 5

Output:-
The sum of all the edge weights: 5