Practical-9: Implementation of Ford-Fulkerson algorithm

```
#include <iostream>
#include <limits.h>
#include <string.h>
#include <queue>
using namespace std;

#define V 6

bool bfs(int rGraph[V][V], int s, int t, int parent[])
{
	bool visited[V];
	memset(visited, 0, sizeof(visited));

	queue <int> q;
	q.push(s);
	visited[s] = true;
	parent[s] = -1;

	while (!q.empty())
	{
		int u = q.front();
		q.pop();

		for (int v=0; v<V; v++)
		{
			if (visited[v]==false && rGraph[u][v] > 0)
			{
				q.push(v);
				parent[v] = u;
				visited[v] = true;
			}
		}
	}
	return (visited[t] == true);
}

int fordFulkerson(int graph[V][V], int s, int t)
{
	int u, v;


	int rGraph[V][V];
	for (u = 0; u < V; u++)
		for (v = 0; v < V; v++)
			rGraph[u][v] = graph[u][v];

	int parent[V];

	int max_flow = 0;
	while (bfs(rGraph, s, t, parent))
```

```
        {
                int path_flow = INT_MAX;
                for (v=t; v!=s; v=parent[v])
                {
                        u = parent[v];
                        path_flow = min(path_flow, rGraph[u][v]);
                }
                for (v=t; v != s; v=parent[v])
                {
                        u = parent[v];
                        rGraph[u][v] -= path_flow;
                        rGraph[v][u] += path_flow;
                }
                max_flow += path_flow;
        }
        return max_flow;
}
int main()
{

        int graph[V][V] = { {0, 16, 13, 0, 0, 0},
                                                {0, 0, 10, 12, 0, 0},
                                                {0, 4, 0, 0, 14, 0},
                                                {0, 0, 9, 0, 0, 20},
                                                {0, 0, 0, 7, 0, 4},
                                                {0, 0, 0, 0, 0, 0}
                                        };

        cout << "The maximum possible flow is " << fordFulkerson(graph, 0, 5);

        return 0;
}
```

Output: