NAME:	Eshan Bhuse
UID:	2021300013
SUBJECT	Design and analysis of algorithm
EXPERIMEN TNO:	9
AIM:	Approximation algorithms (The vertex-cover problem)
ALGORITHM:	1) Initialize the result as {} 2) Consider a set of all edges in given graph. Let the set be E. 3) Do following while E is not empty a) Pick an arbitrary edge (u, v) from set E and add 'u' and 'v' to result b) Remove all edges from E which are either incident on u or v. 4) Return result

PROGRAM:

```
#include<iostream>
#include <list>
using namespace std;

class Graph
{
int V;
list<int> *adj;
public:
Graph(int V);
void addEdge(int v, int w);
void printVertexCover();
};

Graph::Graph(int V)
{
this->V = V;
adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
adj[v].push_back(w);
adj[w].push_back(v);
}


void Graph::printVertexCover()
{

bool visited[V];
for (int i=0; i<V; i++)
visited[i] = false;

list<int>::iterator i;


for (int u=0; u<V; u++)
{

```

```
{

if (visited[u] == false)
{

for (i= adj[u].begin(); i != adj[u].end(); ++i)
{
int v = *i;
if (visited[v] == false)
{

visited[v] = true;
visited[u] = true;
break;
}
}
}
}


for (int i=0; i<V; i++)
if (visited[i])
cout << i << " ";
}

```

```
}


int main()
{
int n,m,i,a,b;
cout<<"\nEnter number of vertices : ";
cin>>n;
cout<<"\nEnter number of edges : ";
cin>>m;
Graph g(n);
for(i=0;i<m;i++)
{
    cout<<"\nEnter edge "<<(i+1)<<" : ";
    cin>>a;
    cin>>b;
    g.addEdge(a,b);
}

g.printVertexCover();

return 0;
}

```

```
Enter number of vertices : 5

Enter number of edges : 6

Enter edge 1 : 0 1

Enter edge 2 : 1 2

Enter edge 3 : 2 3

Enter edge 4 : 3 0

Enter edge 5 : 2 4

Enter edge 6 : 3 4
0 1 2 3

...Program finished with exit code 0
Press ENTER to exit console.
```

OBSERVATION:	Thus, we observe that by using the approximate algorithm it provides us a solution to the vertex cover problem that is not guaranteed to be optimal, but it provides a good approximation with a small running time. In particular, the algorithm guarantees that the size of the vertex cover returned is at most twice the size of the optimal vertex cover. This makes the algorithm useful for large graphs where finding the optimal solution may be computationally infeasible. In our example, the program did not provide us with the correct answer but gave us a good approximation of how the optimal answer would look like which would be feasible where determining the optimal cover would be nearly impossible.
CONCLUSION:	Thus, after performing this experiment I understood and implemented the vertex cover problem using the approximation algorithm.