

```

void Graph::BFS(int s) {
    bool* visited = new bool[V];
    for (int i = 0; i < V; ++i)
        visited[i] = false;

    queue<int> Q;
    vector<int>::iterator it;

    Q.push(s);
    visited[s] = true;

    while (!Q.empty()) {
        s = Q.front();
        Q.pop();

        for (it = adj[s].begin(); it != adj[s].end(); ++it) {
            if (!visited[*it]) {
                Q.push(*it);
                visited[*it] = true;
                cout << s << "->" << *it << endl;
            }
        }
    }
}

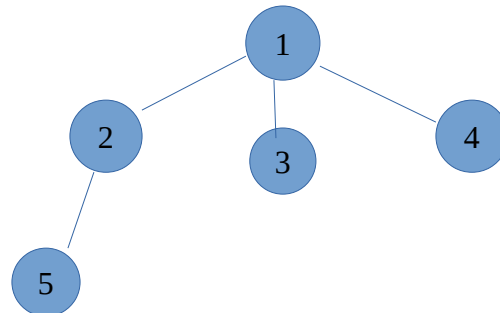
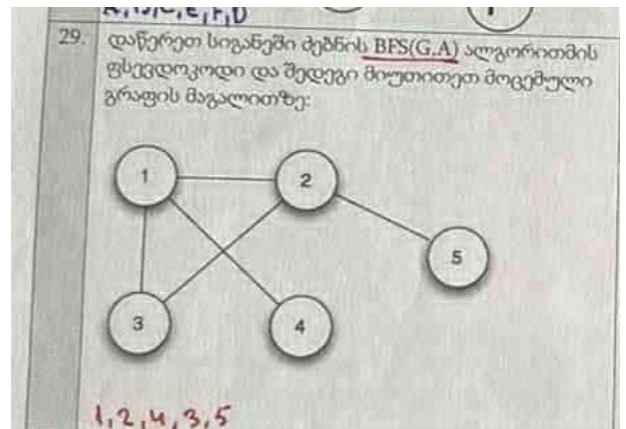
```

visit

1	2	3	4	5
1	1	1	1	1

Queue

1	2	3	4	5
1	2	3	4	5

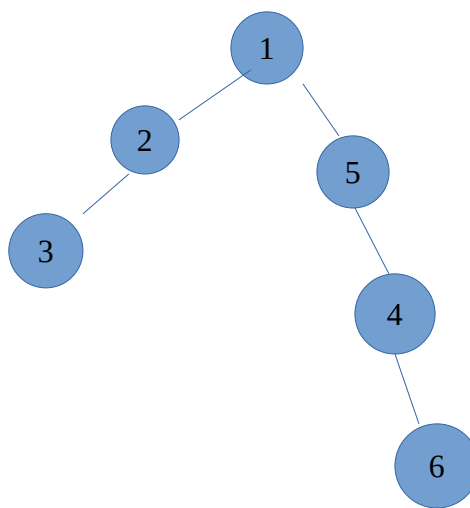
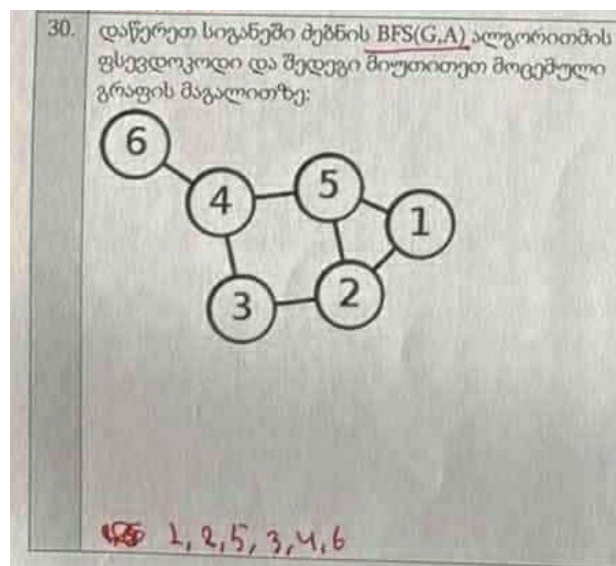


30.
visit

1	2	3	4	5	6
1	1	1	1	1	1

Queue

1	2	5	3	4	6
---	---	---	---	---	---



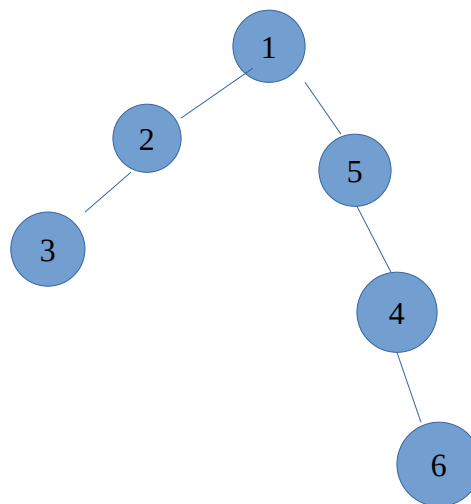
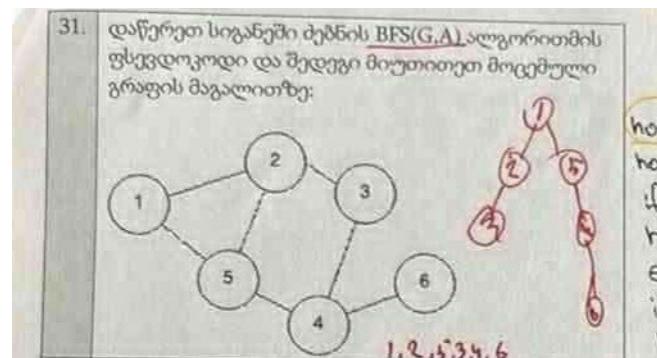
31.

visit

1	2	3	4	5	6
1	1	1	1	1	1

Queue

1	2	5	3	4	6
---	---	---	---	---	---



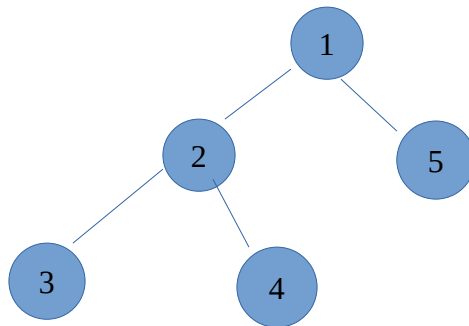
32.

visit

1	2	3	4	5
1	1	1	1	1

Queue

1	2	5	3	4
---	---	---	---	---



32. დაწერეთ სიგანეში ძებნის BFS(G,A) ალგორითმის ფსევდოკოდი და შედეგი მოუთხოვეთ მოცემული გრაფის მაგალითზე:

Handwritten notes for problem 32:

- 3 2 4 1 5
- 1, 2, 5, 3, 4
- 1, 2, 5, 3, 4

33. აღწერეთ ბინარული ძებნის ხეში მონაკვეთის ჩასმის ალგორითმი და დაწერეთ INSERT node *root, int

```

graph TD
    1((1)) --- 2((2))
    1 --- 5((5))
    2 --- 3((3))
    2 --- 4((4))
  
```