1 Q4

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

struct Node {
int value;
Node* next;
Node(int val, Node* next): value(val), next(next)
{}
};
Node* merge(Node *u, Node *v) {

    if (u == nullptr){
        return v;
    } else if ( v == nullptr ) {
        return u;
    }
    else {
        Node* temp = new Node(v->value, merge(u->next, v->next));
        u->next = temp;
        return u;

    }

}

int main(){

   Node* seq1 = new Node(1, nullptr);
   seq1->next = new Node(2, nullptr);
   seq1->next->next = new Node(3, nullptr);

   Node* seq2 = new Node(2, nullptr);
   seq2->next = new Node(4, nullptr);
   seq2->next->next = new Node(6, nullptr);

    Node* merged_list = merge(seq1, seq2);
    while (merged_list) {
        cout << merged_list->value;
        merged_list = merged_list->next;
    }
    cout << endl;

   seq1 = new Node(4, nullptr);
   seq1->next = new Node(2, nullptr);
```

```
  seq2 = new Node(4, nullptr);
  seq2->next = new Node(3, nullptr);
  seq2->next->next = new Node(2, nullptr);
  seq2->next->next->next = new Node(1, nullptr);

   merged_list = merge(seq1, seq2);
   while (merged_list) {
       cout << merged_list->value;
       merged_list = merged_list->next;
   }
   cout << endl;



  return 0;

}
```

2 Q6

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
  int n, m;
  cin >> n >> m;
  map<string, GraphNode*> nodes;
  for (int i = 0; i < n; i ++)
  {
    string nodename;
    cin >> nodename; // O(1)
    nodes[nodename] = new Node(nodename); // O(log n)
  } // O(n log n + n)
  map<string, GraphNode*>::iterator it_start;
  map<string, GraphNode*>::iterator it_end;
  for (int j = 0; j < m; j ++)
  {
    string nodenameStart, nodenameEnd;
    cin >> nodenameStart; // O(1)
    cin >> nodenameEnd;   // O(1)
    it_start = nodes.find(nodenameStart); // O(log n)
    it_end = nodes.find(nodenameEnd);     // O(log n)
    (*it_start)->addEdgeTo(*it_end);      // O(1)
    (*it_end)->addEdgeFrom(*it_start);    // O(1)
    } // O(m log n + m) -> O( (m + n) log n )
   // other stuff using the graph goes here - not yours to worry about
}
```