For each of the following procedures, enter the big-O of its running times. Please represent v's size as m, and express your big-O as a function of n and m.

Question 1:

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

void a(vector <int> &v, int n)
{
  int i;
  for (i = 0; i < n; i++) {
    v.push_back(i*i);
  }
}
```

Question 2:

```
#include <vector>
#include "nlohmann/json.hpp"
using namespace std;
using nlohmann::json;

string vtojs(vector <string> &v)
{
  json j;
  size_t i;

  for (i = 0; i < v.size(); i++) j[v[i]] = i;
  return j.dump(2);
}
```

Please assume that all of the strings in v are less than 10 characters.

Question 3:

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

bool b(vector <int> &v, int n)
{
  multiset <int> s;
  int i;

  for (i = 0; i < v.size(); i++) {
    s.insert(v[i]);
  }
  return (s.find(n) != s.end());
}
```

Question 4:

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

int find(vector <int> &v, int n)
{
  int l, h, m;

  l = 0;
  h = v.size();

  while (l < h) {
    m = (l+h)/2;
    if (m == h) m--;
    if (v[m] == n) return m;
    if (v[m] > n) {
      h = m;
    } else {
      l = m+1;
    }
  }
  return l;
}
```

Clicker Question Answers

- Question 1: `push_back()` is *O(1)* and the loop iterates *n* times: *O(n)*.
- Question 2: Recall, JSON objects are implemented with maps. This is why they are sorted when you print or **dump()** them. Therefore, this is *O(m log m)*.
- Question 3: The loop iterates *m* times, and the `insert()` operation on average is *O(log m)*. The last `find()` is *O(log m)*, so it may be ignored. The answer is *O(m log m)*.
- Question 4: The variable *m* is set to the midpoint between *l* and *h*. Thus, *(h-l)* is reduced by half at each iteration. *(h-l)* starts at *m*, so this loop iterates *O(log m)* times. The answer is *O(log m)*.