Clicker Questions

- **Question 1**: What is the big-O running time of **p1.cpp**?
- **Question 2**: What is the big-O running time of **p2.cpp**?
- **Question 3**: What is the big-O running time of **p3.cpp**?
- **Question 4**: When we compile and run **p1.cpp**, how many bytes does the program consume while it's running?

Please use the following multiple choice answers:

- A: *O(1)*
- B: *O(log n)*
- C: *O(i)*
- D: *O(n)*
- E: *O(1 + n)*
- F: *O(n log i)*
- G: *O(n log n)*
- H: *O(n + n log n)*
- I: *O(n*i)*
- J: $O(n^2)$

p1.cpp

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

int main()
{
  int i, n;
  map <int, int> m;

  cin >> n;
  for (i = 0; i < n; i++) {
    m[i] = n;
  }
  return 0;
}
```

p2.cpp

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

int main()
{
  int i, n;
  map <int, int> m;

  cin >> n;
  for (i = 0; i < n; i++) {
    m[n] = i;
  }
  return 0;
}
```

p3.cpp

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

int main()
{
  int i, n;
  map <int, int> m;
  map <int, int>::iterator mit;

  cin >> n;
  for (i = 0; i < n; i++) {
    mit = m.find(n);
    if (mit != m.end()) {
      m.erase(mit);
    } else {
      m.insert(make_pair(n,i));
    }
  }
  return 0;
}
```

Clicker Answers

- **Question 1**: *G: O(n log n)* -- *n* insertions of different numbers into an empty map.
- **Question 2**: *D: O(n)* -- Since maps do not support duplicate elements, this map's size is always 1.
- **Question 3**: *D: O(n)* -- Here, we either insert *n* into the map, or we erase it. In either case, the map's size is either 0 or 1, so the operations on it are *O(1)*. There are *O(n)* operations, so the answer is *O(n)*.
- **Question 4**: *D: O(n)* -- A map with *n* elements is implemented as a balanced binary tree. Trees with *n* elements consume *O(n)* bytes -- there are *n* nodes and each node has *O(1)* links.