All of the questions refer to this procedure:

```
int find_em(const vector <int> &r,
            const map <int, int> &p,
            int n)
{
  int found;
  int i, index;

  found = 0;
  for (i = 0; i < n; i++) {
    index = rand() % r.size();
    if (p.find(r[index]) != p.end()) found++;
  }
  return found;
}
```

Question 1: Assume that the size of **r** is o, and the size of **p** is m. What is the running time of this function? Please choose from the following multiple choice answers:

A. $O(m \log(m))$
B. $O(m \log(n))$
C. $O(m \log(o))$
D. $O(n \log(m))$
E. $O(n \log(n))$
F. $O(n \log(o))$
G. $O(o \log(m))$
H. $O(o \log(n))$
I. $O(o \log(o))$

Question 2: Now assume that the size of **r** is m, and that the size of **p** is $m/2$. When I run this procedure with n = 10,000,000 and m = 1,000, it takes 2.8 seconds to run. Roughly how long will it take when n = 20,000,000 and m = 1,000? (The units will be "seconds").

Question 3: Given the same assumptions as in Question 2, roughly how long will it take when n = 10,000,000 and m = 2,000?

Clicker Question Answers

Question 1: The loop iterates n times, and since the map is size m, each **find()** operation is $O(log(m))$. So the answer is D: $O(n\ log(m))$.

Question 2: Since the loop is $O(n\ log(m))$, we expect the time to double when we double n. The answer is 5.6.

Question 3: The answer is a little trickier here. We want to assess the impact of $log(2m)$ in $O(n\ log(m))$. Since the base of the logarithm is two, $log(2m)$ is equal to $log(m)+1$. So the impact is not going to be great. Were I estimating, I'd say that $log(1000)$ is roughly 10 (remember your powers of two -- 1024 is 2^{10}), which means that $log(2000)$ is roughly 11. So I would multiply 2.8 by 11/10 to get 3.08. I accepted any answer between 2.8001 and 3.5.

BTW, when I programmed this up and tested, the running times matched our estimates!