

## 1) (5 pts) ALG (Algorithm Analysis)

Consider the following function with integer inputs n and m:

```
void solveit(int* array, int n, int m) {  
  
    int i, res = 0;  
    for (i=0; i<n; i++) {  
        int low = 0, high = m;  
        while (low < high) {  
            int mid = (low+high)/2;  
            if (f(mid) < array[i])  
                low = mid+1;  
            else  
                high = mid;  
        }  
        printf("%d\n", low);  
    }  
}
```

You may assume that the function f that is called from solveit defines a monotonically increasing function that runs in  $O(1)$  time. With proof, determine the run-time of this function in terms of n and m.

The outer loop runs n times. The inner loop runs a binary search between 0 and m, subdividing that interval by 2 with each loop iteration. The most number of times this loop could run is roughly  $\lg m$ . (A more formal proof simply let's k equal the number of times the loop runs and solves for k in the equation  $\frac{m}{2^k} = 1$ . Multiply both sides to get  $m = 2^k$ . By definition of log,  $k = \log_2 m$ .) It follows that the run time is  $O(n \lg m)$ .

**Grading: 1 pt for outer loop, 3 pts for inner loop, 1 pt for multiplying. (Note: to award the full three points, student must either point out that code is a binary search or that repeated division by 2 is going on to the interval high-low. They don't need to set up the equation with k, the number of iterations the loop runs.)**