CSE 143 Au00 Additional Problems for Sections
November 28, 2000

For each of the following sequences of code, use *O*( ) notation to describe the running time in terms of N. Give a lower bound for the running time; don't just answer that all are $O(2^n)$.

1.
```
for (int j = 1; j <= N; j++) {
   int k = 1;
   while (k < j)
      k = k * 2;
   cout << j << "  " << k << endl;
}
```

Running time is *O*(____n log n_____)

Followup: Suppose the statement `k=k*2;` were replaced with `k=k*3;` . How would this change the running time, if any? No change; still O(n log n).

What do you think this code fragment does? For each integer from 1 to N, prints that integer and the smallest power of 2 greater than or equal to that integer.

2.
```
double avg(double m[N][N], int r, int c) {
   double sum = 0.0;
   for (int j = r-1; j <= r+1; j++)
      for (int k = c-1; k <= c+1; k++)
         sum = sum + m[j][k];
   return sum/9.0;
}
```

```
double puzzle[N][N];
...
for (int r = 1; r < N-1; r++)
   for (int c = 1; c < N-1; c++)
      puzzle[r][c] = avg(puzzle,r,c);
```

Running time is *O*(____n^2_____)

Followup: Does it make any difference whether `puzzle` is or is not copied each time the function is called? (i.e., Does it matter if the parameter mechanism is call-by-value instead of call-by-reference?) Yes. If the array is not copied, each function call requires constant time, and there are $O(n^2)$ function calls. If the array were copied, then each function call would require $O(n^2)$ work by itself. Since there are $O(n^2)$ calls, the total work is $O(n^2)$ times $O(n^2)$, or $O(n^4)$.