Assignment 1

Due: Tuesday, January 27, 2015 (hand in before class)

0) (2 pts.) Housekeeping tasks:

1. Visit the course website at http://www.cs.purdue.edu/homes/seh/381webSp15/. Follow the links and read the information provided. Set appropriate bookmarks needed during the semester.

2. Sign up to Piazza at piazza.com/purdue/spring2015/cs381. Material will be posted on Piazza. Piazza also serves as the discussion forum of topics and questions of interest to the entire class.

3. Register your i-Clicker on Blackboard. Use the same registered clicker during the semester.

1) (8 pts.) Partition the following functions representing running times into equivalence classes so that \( f(n) \) and \( g(n) \) are in the same class if and only if \( f(n) = \Theta(g(n)) \). Rank the classes from smallest to largest (in terms of growth rate with respect to \( n \)). Logarithms are base 2 unless stated otherwise. You only need to give the final ordering from small to large with clearly indicating functions in the same equivalence class.

<table>
<thead>
<tr>
<th>Function</th>
<th>( 3n \log \log n )</th>
<th>( 10n^3 + 14n^2 \log^6 n )</th>
<th>( 10n \sqrt{n \log n} )</th>
<th>( 4\log^3 n )</th>
<th>( (\sqrt{2})^\log n )</th>
<th>( \sqrt{n} + (\log n)^3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 8\sqrt{n} + 6 \log^6 n )</td>
<td>( \left( \frac{n}{\log n} \right)^3 )</td>
<td>( 81 \log n^3 )</td>
<td>( 3n \log n )</td>
<td>( \sqrt{\log n^2} )</td>
<td>( 3n + 5\sqrt{n} )</td>
<td></td>
</tr>
</tbody>
</table>

2) (8 pts.) Review proofs by inductions. In your proofs, be precise and show all needed steps.

1. Use induction to show that \( n! > 2^n \) for \( n > 3 \).

2. Use induction to show that \( \sum_{i=1}^{n} (2i - 1) = n^2, \ n \geq 1 \).

3) (22 pts.) Given is an array \( A \) containing \( n \) distinct integers in arbitrary order. Describe and analyze an efficient algorithm for each of the following problems. Clearly state the achieved running time in big-O and \( \Theta \)-notation. Do not give code, but explain your solution in a clear and precise manner. In addition to the running time, argue the correctness of your algorithm.

1. Determine the maximum difference between any two elements in \( A \). For example, for \( A = [2, 10, 12, 4, -9, 0, -5, 8, 1, -7] \) the answer is 21 (difference between -9 and 12).

2. Determine the number of elements between two given values \( x \) and \( y \) with \( x < y \) (i.e., the number of elements \( z \) such that \( x < z < y \)).

3. Determine the 5-th largest element in array \( A \).

4. Given is also an integer \( r \), \( r > 0 \). Determine two elements \( A[k] \) and \( A[p], k < p \), such that \( p - k \leq r \) and \( |A[p] - A[k]| \) is a maximum. For example, for \( r = 2 \) and \( A = [-8, 3, 2, -5, 6, 10, 14] \) the answer is \( A[k] = -5 \) and \( A[p] = 10 \) achieving a difference of 15.
4) (10 pts.) How many times is function \( F \) called in each code segment given below when \( n = 2^r \)? Clearly explain your answer and express bounds in terms of \( n \) in big-O and \( \Theta \)-notation. Review the pseudocode convention in Section 2.1 (if needed).

**Code Segment 1**

```
for i = 1 to n do
  j = 1
  while j ≤ n do
    j = 2^i
    for k = 1 to j do
      F(i,j,k)
```

**Code Segment 2**

```
while n > 1 do
  for i = 1 to n do
    F(i,n)
  n = n/4
```