Assignment 8, Time Complexity

Due: WED, Dec 4th, by 11:59 p.m. (Late penalty 5% per hour through 5:00 p.m., Dec 5th)

> Directions: You may not work with others on this assignment—if you have questions or need help, ask me. Any violation of the instructions for a given problem will result in 0 points for that question. You must write "main" for Problems 1 – 3. Put your answer to Problem 4 in the same java file as your program for Problem 5. You don't need to test the method of Problem 4, but it must compile. Turn in your output for Problems 1 – 3 and Problem 5.

Problem 1 Write method *evaluateAsst* which evaluates the formula F for the given truth assignment.

```
boolean evaluate(int n, int k, int [][] F, boolean [] truthAsst)
```

Problem 2 Write the method below which takes a boolean formula *F* with *n* variables and *k* clauses (each clause has exactly 3 literals) and returns an undirected graph *W* such that *W* has a *k* clique if and only if *F* is satisfiable. The method must run in polynomial time in terms of *k* = the size of the input. (*Why is **k** the size of the input?*)

```
int [][] reduce_3SAT_to_CLIQUE(int n, int k, int [][] F)
```

Problem 3 For each output from Problem 2, use your *kClique* method from Assignment 7 to find a *k*-clique (if one exists). If there is a *k*-clique, pass this *k*-clique to the method below (write the method below), which will return a boolean array of length *n* containing a satisfying assignment to the formula *F* from which the graph was created.

```
boolean [] getAsstFromClique(int n, int k, int [] kClique, int [][] F)
```

Now use your method *evaluate* from above to verify that the boolean assignment returned actually satisfies the formula *F* (makes it evaluate to true).

Problem 4 Write the method below which takes an instance of the SOS problem (an array *A* of *n* integers and a goal *g*) and returns an array *B* of integers (not necessarily the same length as) *A* such that the set *B* can be partitioned if and only if there is a subset of *A* whose elements sum to *g*. Your method must run in polynomial time in *n*. ~~both *n* and *lg W*.~~

```
int [] reduce_SOS_to_PARTITION(int n, int [] A, int g)
```

Problem 5 Write the method below which finds the minimum element and the maximum element of an array using at most $(\mathbf{3}\boldsymbol{n}/\mathbf{2}\ \ -\mathbf{2})$ comparisions, where *n* is the length of *A* (and *n* is even). It returns an array of size 2, where the first element is the minimum value in *A* and the second is the maximum value in *A*. Test your method on several arrays of even length.

```
int [] minMax(int n, int [] A)
```