CS 18200 Programming Assignment 1 (100 Points Possible)
Fall 2015 Due on BLACKBOARD, [Monday, Oct. 19th at 11:59pm]

Late Penalty:

- 0 to 24 hours late: 25% off
- 24+ to 48 hours late: 50% off
- 48+ hours late: no credit

Be sure to read the course policies posted on Blackboard, especially the bullets on the quality of your write-up, acknowledging collaborators and sources, and what is considered cheating.

The Boolean Satisfiability Problem

Let P be a propositional formula, and let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of variables in P. For each variable v_i, let a_i be a truth value (either **true** or **false**) assigned to that variable. Then $A = \{a_1, a_2, \ldots, a_n\}$ is a truth assignment of truth values to the variables in V. The Boolean Satisfiability Problem is, given a formula P, to determine whether there exists a truth assignment A such that when $v_i = a_i$ for all i, P is **true**.

For example, if P is $(v_1 \land \neg v_1)$, then there is no satisfying assignment such that P is **true**. On the other hand, if P is $((v_1 \lor \neg v_2) \land (\neg v_1 \lor \neg v_3))$, then there is at least one satisfying assignment, namely v_2 is **false** and v_3 is **false**.

In this assignment, you will:

- implement your own solver for the **Boolean Satisfiability Problem**
- write this solver in the **Java Programming Language**
- use a **recursive** algorithm that employs exhaustive search over all truth assignments to solve this problem

The goal of this project is to familiarize you with recursive programming solutions that uses exhaustive search. At each level of the recursion, the recursive procedure should set the next Boolean variable to **true**, then recurse, and upon return reset that Boolean variable to be **false** and recurse again. When the procedure reaches a level of recursion at which all variables have been assigned a truth value, then the can be evaluated to be either **true** or **false**. If it is **true**, then you can return the satisfying assignment of variables. If the assignment is not satisfying, then upon return to a level at which a variable was assigned **true**, change the assignment of that variable to be **false** and recurse down again.

Program Input

The input for your program will be as follows:

```
Problem Name
n m
clause 1
⋮
clause m
```

The first line specifies the problem name. The line following gives two numbers, `n` and `m`, where `n` is the number of Boolean variables the problem will have and `m` is the number of clauses the problem will have. Our formulas will be in Conjunctive Normal Form (CNF for short). IN CNF, each formula will be the conjunction of clauses, where a *clause* is a disjunction of literals, and each *literal* will be either a variable or the negation of a variable. Each problem will have a total of `m + 2` lines, and a single input file can have any number of problems in it.

Here is an example of such an input file:

```
Problem 1
2 1
-1 2
Problem 2
3 2
2 1 3
-2 -1
```

Interpretation of the Input

Here is the breakdown of `Problem 2` from above.

```
Problem 2      // NAME OF PROBLEM
3 2            // 3 variables (e.g., $x_1, x_2, x_3$ ), 2 clause
-2 1 3         // 1st clause: ( $\neg x_2 \lor x_1 \lor x_3$ )
-2 -1          // 2nd clause: ( $\neg x_2 \lor \neg x_1$ )
```

With the above breakdown, once the input is parsed, you have a number of clauses and variables. First, note that the numbers in each clause correspond to a specific variable (with negatives to represent the negation of that variable). So `1` is interpreted as x_1, `-9` is interpreted as $\neg x_9$, and so on. Note that the variable numbering starts at **1** and ends at the number of variables specified. So for example, if we have 3 variables, there will only be the numbers 1, 2, or 3 appearing in the clauses below.

The final step in the interpretation of the input is to evaluate whether the following Boolean expression is `SATISFIABLE`:

$$(\neg x_2 \lor x_1 \lor x_3) \land (\neg x_2 \lor \neg x_1)$$

Your Task

Your overall task for this assignment:

- implement your own solver for the **Boolean Satisfiability Problem**
- write this solver in the **Java Programming Language**
- use a **recursive** algorithm employing exhaustive search to solve this problem

More specifically, here is a breakdown of what you are expected to complete for this assignment.

1. Parsing the Input and Output Format (15 points)

You are expected to implement a parser that can read the input as specified above. Your parser should be able to handle an arbitrary number of problems from a single input. For example, your program should be able to read

```
Problem 1
2 1
-1 2
Problem 2
3 2
2 1 3
-2 -1
```

and evaluate `Problem 1` followed by `Problem 2` and so on. You are to use the *System.in* call to read the input.

You can safely assume that all input to your program will be correct as described in the **Program Input** section. You can also assume that there will always be **at least 1 variable and clause** for your program to evaluate.

The format of your program output must be as follows:

- Print the name of the problem (i.e., the first line of a problem from the input file) followed by a colon (`:`), followed by the number of variables and the word `Variable(s)`, followed by the number of clauses and the word `Clause(s)`

- Print whether or not the program is `Satisfiable`
- If the problem **is satisfiable**, print the truth assignment you found that makes it satisfiable in variable order. So if you have 3 variables, print the truth assignments of x_1, x_2, x_3 in that order.

Here is an example to help illustrate what the output must look like:

Given this input:

```
Test 1
3 3
-1
-2
3
Test 2
1 2
1
-1
Test 3
5 8
5 -4 -2
1 3 -2
-5 1 4
-4 -2 -3
-3 4 5
1 -2 5
3 -2 5
2 3 -4
Test 4
3 5
1 2 3
2 -3
1 -2
-1 -3 -2
3 -1
```

The output for this input should be:

```
Test 1: 3 Variable(s) 3 Clause(s)
Satisfiable
false false true
```

```
Test 2: 1 Variable(s) 1 Clause(s)
Unsatisfiable
Test 3: 5 Variable(s) 8 Clause(s)
Satisfiable
true true true false true
Test 4: 5 Variable(s) 6 Clause(s)
Unsatisfiable
```

Again, your program must be able to evaluate multiple problems from one input file and output results from each problem.

You are to use *System.out* to print all outputs of your program.

Please follow the output format guidelines **exactly** as they are written. If you do not, you will lose points.

2. Correct Solutions (25 Points)

We will test the correctness of your program's output.

If a problem in the input is **unsatisfiable**, any other answer in the output will receive zero points for that problem.

If a problem in the input is **satisfiable**, we will test if your truth assignments satisfy the Boolean expression the problem represents. If your assignment does not satisfy that expression, you will receive zero points for that problem.

3. Recursion (55 Points)

The main goal of this project is to give you experience implementing a recursive algorithm that uses exhaustive search. So you are required to implement your solution with recursion. **If you do not implement a recursive procedure, you will receive zero points for the entire assignment**.

More specifically, you are to assign **true** to a variable and then recurse on the Boolean expression to make another truth assignment to the next variable. You continue until you have a complete truth assignment and then test whether that truth assignment either **satisfies** the expression, or fails to satisfy the expression. If it fails, return up to the nearest level of recursion in which the variable set at that level was set to **true**, set it to **false**, and recurse from there.

4. Program Structure (5 Points)

Your `main` function must be written in a class called `Main` and in a file called `Main.java`. Other than that, you are free to implement any additional classes, functions, etc. to help you solve the problem.

Deliverables

You are to turn in all the required `.java` files to run your program on **Blackboard**. On the CS 182 Blackboard page, click on the tab titled **Programming Assignment 1 Submission** to submit your files. You must put all of these files in a `.zip` folder named as follows:

`LastName-FirstName-Project1.zip`

Additionally, you must put your First and Last name at the top of each of your `.java` files as comments.

All files are due by 11:59pm on Monday, October 19th.

Additional Notes

- You are not allowed to use any external Java libraries or packages for this project. Your program must be implemented using default Java packages and libraries. **If you use an external library or package, you will receive a zero for this project.**
- You must do this project individually. **Sharing of code between students or using code found online is strictly prohibited**. A violation of this policy is considered an act of **Academic Dishonesty**.