CSE 3401 Assignment 1
Winter 2013

Date out: January 21. Date due: February 10, at 11:30 pm

- The submitted assignment must be based on your individual work. Review the Academic Honesty Guidelines for more details.
- This assignment constitutes 6% of your total mark for the course and is marked out of 130.
- You should adhere to the coding guidelines posted on the website; comment your code and test it thoroughly.
- You should not use any built-in predicates in your solutions except for (`not/1`, `member/2`, `append/3`, `is`, arithmetic operators, comparison operators, logical symbols and the cut (`!`)). If you need to use any other built-in predicates not mentioned in the above list, you can confirm it with the instructor.
- To define the predicates in Section 1, you may define some auxiliary relations if that helps in defining the required predicates.
- You should Submit 3 files for this assignment:
  - a1.pl, which is the source code of your solutions for Section 1. This file should follow the Prolog solutions format provided on the website (solutionFormat.txt).
  - a1.txt, which consists of the test cases you have used to test each question in Section 1, as well as the results of testing. Include a header with your name, student number and cse login.
  - a1.pdf, which consists of your answers to Section 2. Include a header with your name, student number and cse login.

  Soft Copies: Gather all the required files in a directory named `a1answers` and submit it electronically by the deadline. To submit electronically, use the following Prism lab command:

  ```
  submit 3401 a1 a1answers
  ```

  Alternatively, you may use web submit (https://webapp.cse.yorku.ca/submit/) and choose the correct course and assignment number to upload your files.

Section 1: Prolog Programs (70 points)

Q1. Define a predicate **`maxNumber(N1,N2,MaxNum)`** that takes as input 2 integers **`N1`** and **`N2`**, and returns the larger number in **`MaxNum`**, for example:

```
?-maxNumber(1,2,MaxNum).
MaxNum=2.
```

Q2. We have the following information about children's book authors:

Andersen was born in 1805 and died in 1875.
Dodgson was born in 1832 and died in 1898.
Aardema was born in 1911 and died in 2001.
Twain was born in 1835 and died in1910.
Travers was born in 1899 and died in 1986.

a. Represent the above as a set of facts in a knowledgebase.
b. Define a predicate **`contemporary(Author1,Author2),`** such that it succeeds (returns True) if the range of years in which Author1 and Author2 were alive overlap. For example:

```
?-contemporary(andersen,dodgson).
True.

?-contemporary(andersen,aardema).
False.
```

c. Define a predicate **`aliveIn(Author, Year)`** which succeeds if **`Author`** was alive during the year indicated by **`Year`**, for example:

```
?-aliveIn(andersen,1806).
True.
```

Q3. Define a predicate **`rotateListElement(List1, List2),`** which takes **`List1`** as its first argument and returns **`List2`** which is the result of moving the first element of **`List1`** to the end, for example:

```
?-rotateListElement ([1, 2, 3], List2).
List2=[2,3,1].
```

Q4. Define a predicate **`highestGrade(List, MaxGrade),`** which takes **`List`** as its first argument, which is a list of students names and grades as **`st(Name, Grade),`** and returns, the highest grade in the list, for example,

```
?-highestGrade([st(john,78),st(anne, 90),st(paul, 68)], MaxGrade).
MaxGrade=90.
```

Q5. Define a predicate **`listToNumber(List, ListValue)`**, which takes **`List`** which is a list of integers **,** all between 0 and 9 inclusive, and whose second argument, **`ListValue`**, should become instantiated to an equivalent number, treating the list of digits as denoting a decimal value. For example,

```
?-listToNumber([1,9,0,0,2], N).
N=19002.
```

Q6. Define a predicate **`countListElements(List, Count)`**, which takes **`List`** as input, and its second argument should become instantiated the count of total of atomic elements which are not lists. For example,

```
?- countListElements([[],a,2,[1,2]], Count).
Count = 2.

?-countListElements([[],[1,2]], Count).
Count = 0.
```

Q7. Define a predicate **`flattenList(List1, List2)`**, where **`List1`** can be a list of lists and/or atomic elements, and **`List2`** is flattened so that the elements of **`List1`**'s sublists (to any depth) are returned as a plain list, for example:

```
?- flattenList([[1],a,2,[],[1,2,[[3]]]], List2).
List2 = [1,a,2,1,2,3].
```

Section 2: Artificial Intelligence (60 points)

Mary, John and Paul are students at a small college. Every student of the college who is not a Chemistry major is a Physics major. Physics majors do not like Biochemistry. Anyone who does not like Statistics is not a Chemistry major. Mary dislikes whatever Paul likes and likes whatever Paul dislikes. Paul likes Biochemistry and Statistics.

1. Write sentences in First Order Logic that represent the above knowledge. Also, include a glossary where you explain the intended meaning of each predicate, function and constant symbol in English.

2. Convert these sentences to clausal form and give the resulting set of clauses

3. Use resolution to prove that there exists a student of the college who is a Physics major but not a Chemistry major. Show how the query is represented in First Order Logic and which clause is added to the theory. Provide the complete resolution derivation – In sequence or tree format – indicating which literals / clauses are resolved and the unifier used.

4. Suppose you had been told that Mary likes whatever Paul dislikes, but you had not been told that Mary dislikes whatever Paul likes. Prove that the resulting set of sentences does not logically entail that there is a student of the college who is a Physics major but not a Chemistry major. You can prove this by providing an interpretation that satisfies the knowledgebase but falsifies the query.