Exercises 2 – 5 are programming problems. Submit them in a single file named `a13.pl` electronically per the instructions for your course.

1. Study Bratko, Sections 6.1 – 6.6, except 6.1.2.

2. Study the article, *Unification: A Multidisciplinary Survey*, Kevin Knight, ACM Computing Surveys, Vol 21, No. 1, March 1989.

   http://www.cslab.pepperdine.edu/warford/cosc450/Unification-Knight.pdf

3. Do Bratko, Exercise 6.3.
   Because `ground/1` is a builtin predicate, you must name your predicate `my_ground`. The following two queries should succeed

   ```
   ?- my_ground( w( x( a) ,y( b), z( c))).
   ?- my_ground( w( x( a) ,b , z( c))).
   ```

   and the following two queries should fail.

   ```
   ?- my_ground( w( x( a) ,y( B), z( c))).
   ?- my_ground( w( x( a) ,B, z( c))).
   ```

   should fail. An atomic term is grounded. A compound term is grounded if each of its arguments is grounded. For compound terms, use `=..` to access the argument list, and write another predicate to test each argument in the list.

4. Do Bratko Exercise 6.7.
   You must use `asserta/1` and `retract/1` and name your predicate `my_copy_term`. Here is a test of `my_copy_term`.

   ```
   ?- my_copy_term( abc(X, def(X), Y), C).

   C = abc(A,def(A),_)
   ```

   You can do this with one rule for `my_copy_term( Term, Copy)`. If you dynamically install a new predicate in the database with `Term` as its argument, then retract that predict with `Copy` as its argument, then `Copy` will be a copy of `Term`.

5. Do Bratko Exercise 6.8.
   Here is the specification of `powerset`.

   ```
   % powerset( Set, P)
   % P is a set of all the subsets of Set
   ```

   Here is a test of `powerset`.

   ```
   ?- powerset( [a,b,c], P).

   P = [[a,b,c],[a,b],[a,c],[a],[b,c],[b],[c],[]]
   ```

Write a predicate `subsets_with_backtracking` that generates the subsets with backtracking, then use `bagof` to collect them all into a list of lists. Here is the specification of `subsets_with_backtracking`.

```
% subsets_with_backtracking( Set, Subset)
% Subset is a subset of Set
```

For example, here is a sample run of `subsets_with_backtracking`.

```
?- subsets_with_backtracking( [a,b,c], Subset).

Subset = [a,b,c] ? ;

Subset = [a,b] ? ;

Subset = [a,c] ? ;

Subset = [a] ? ;

Subset = [b,c] ? ;

Subset = [b] ? ;

Subset = [c] ? ;

Subset = []
```

You can write `subsets_with_backtracking` with one base case fact and two rules, each of which has only one goal. The first rule expresses the fact that the set of all subsets that begin with `a` are the sets with `a` and all the subsets of `[b,c]`. The second rule expresses the fact that `Subset` is a subset of `[a,b,c]` if `Subset` is a subset of `[b,c]`.