4. Logic Programming (Prolog) [8 points]

For this part of the comprehensive assignment, we will ask you to implement part of the RANSAC algorithm following the logic paradigm. You will work with reduced point sets (from part 3) and we ask you to define some of the predicates required to find the dominant plane. You will use the following files:

- `Point_Cloud_1_No_Road_Reduced.xyz`
- `Point_Cloud_2_No_Road_Reduced.xyz`
- `Point_Cloud_3_No_Road_Reduced.xyz`

And the following predicate that reads the point cloud in a file and creates a list of 3D points:

```
read_xyz_file(File, Points) :-
    open(File, read, Stream),
     read_xyz_points(Stream,Points),
    close(Stream).
read_xyz_points(Stream, []) :-
    at_end_of_stream(Stream).
read_xyz_points(Stream, [Point|Points]) :-
    \+ at_end_of_stream(Stream),
     read_line_to_string(Stream,L), split_string(L, "\t", "\s\t\n",
XYZ), convert_to_float(XYZ,Point),
    read_xyz_points(Stream, Points).
convert_to_float([],[]).
convert_to_float([H|T],[HH|TT]) :-
   atom_number(H, HH),
   convert_to_float(T,TT).
```

Predicates to define

- `random3points(Points, Point3)` :
  - This predicate should be true if `Point3` is a triplet of points randomly selected from the list of points `Points`. The triplet of points is of the form `[[x1,y1,z1], [x2,y2,z2], [x3,y3,z3]]`.

- `plane(Point3 , Plane) :`
  - This predicate should be true if `Plane` is the equation of the plane defined by the three points of the list `Point3`. The plane is specified by the list `[a,b,c,d]` from the equation `ax+by+cz=d`. The list of points is of the form `[[x1,y1,z1], [x2,y2,z2], [x3,y3,z3]]`.

- `support(Plane, Points, Eps, N) :`
  - This predicate should be true if the support of plane `Plane` is composed of `N` points from the list of points `Point3` when the distance `Eps` is used.

- `ransac-number-of-iterations(Confidence, Percentage, N) :`
  - This predicate should be true if `N` is the number of iterations required by RANSAC with parameters `Confidence` et `Percentage` according to the formula given in the problem description section.

In this part of the project, you do not have to fully implement the RANSAC algorithm. We simply ask you to create the four predicates listed above. But you must also create test cases for each of your predicate. With Prolog, a test case is generally a predicate that calls the predicate to be tested with all its arguments instantiated.

For example, if you were to write a test case for the predicate `reverse`, one could define the following test case :

```
test(reverse, 1) :- reverse([a,b,c], [c,b,a]).
test(reverse, 2) :- reverse([a], [a]).
test(reverse, 3) :- reverse([], []).
```

And the query :

```
?- test(reverse,N).
```

Note that you do not have to validate the inputs of a predicate, i.e. if a parameter of the predicate has to be a number, you do not have to check what happens if a list is provided instead.

In order to test your predicate, you should use the functions you have created in the other parts of this project. For instance, to obtain the equation of the plane passing by three points, get the results from the corresponding Java, Go or Scheme function and create the Prolog test case from this result.

Testing the `random3points` predicate is tricky, you have to come up with convincing test cases that demonstrates the validity of the results produced by this predicate.

Submit your project in a zip file containing the Prolog file including the test cases.

All your Prolog predicates must have a short header describing what the predicate does and its parameters.