EE 4785 **Homework 2** **Due: 14 February 1996**

For best results, do not start on these problems on or after the night before they are due. Also, submit your own solutions to the problems.

Problem 1: Write a Prolog program so that goal `index(OldList,Element,Index,NewList)` has the following behavior. The goal is issued with `OldList` bound to a list and `Element` bound to an element. If `Element` is in `OldList` then `Index` is matched with the position of `Element` in list `OldList` and `NewList` is matched with `OldList`. If `Element` is not in `OldList`, then `NewList` is matched with a list which consists of `Element` added to `OldList` and `Index` is bound to the index in the new list. The `Element` can be added to the beginning or to the end of the list and `Index` can "count" from the beginning or the end of the list. *However,* behavior must be consistent in that when new elements are added the index of old elements do not change.

For example,

```
 ?- index([a,b,c],a,Index,NewList).

Index = 2
NewList = [a,b,c]

yes
 ?- index([a,b,c],d,Index,NewList),index(NewList,a,Index2,NewerList).

Index = 3
NewList = [d,a,b,c]
Index2 = 2
NewerList = [d,a,b,c]
```

Note that the index of the last element is 0. In the first example, `a` is found at index 2. In the second example `index` is called twice; the first time `d` is added to the list at position 3, the second time `a` is found at position 2. Do not use `assert`, `retract`, or similar predicates.

Problem 2: Write a Prolog program to compress or decompress a list. A list is compressed into a structure of the form `z(Dict,[e(N1,R1),e(N2,R2),...,e(Nx,Rx)])`, where `Dict` is a *dictionary* and `e(Number,Repeats)` describes elements of the list, as described below. A list is compressed by first assigning a number to each distinct element. Elements of the uncompressed list are replaced by `e(Number,Repeats)`, where `Number` is the number assigned to the element and `Repeats` is the number of consecutive times the element appears in that place in the list. Component `Dict` contains the distinct elements, the number of an element corresponds to its position in `Dict`.

For example:

```
 ?- pzip([a,a,b],X).

X = z([a,b],[e(0,2),e(1,1)])
```

Here list `[a,a,b]` is "compressed" into `z([a,b],[e(0,2),e(1,1)])`. The list is uncompressed by issuing goal

```
 ?- pzip(X,z([a,b],[e(0,2),e(1,1)])).

X = [a,a,b]
```

Another example (edited for clarity):

```
 ?- pzip([human(socrates),antidisestablishmentarianism,
     antidisestablishmentarianism,antidisestablishmentarianism,
     foo,foo,1+1,human(socrates),antidisestablishmentarianism],X),pzip(Y,X).

X = z([human(socrates),antidisestablishmentarianism,foo,1+1],
      [e(0,1),e(1,3),e(2,2),e(3,1),e(0,1),e(1,1)])

Y = [human(socrates),antidisestablishmentarianism,
     antidisestablishmentarianism,antidisestablishmentarianism,
     foo,foo,1+1,human(socrates),antidisestablishmentarianism]
```

Do not use `assert`, `retract`, or similar predicates. The solution to the first problem may be used in this problem. Have as much code as possible doing both compression and uncompression.