Problem 1 – Family tree

```
30 ?- mother(brenda,eric).
true.
31 ?- parent(X,eric).
X = fred ;
X = brenda.
32 ?- grandparent(Z,brenda).
Z = helen ;
Z = john ;
Z = claudeen ;
Z = lloyd-sr ;
false.
33 ?- grandmother(W,fred).
W = loraine ;
W = audrey ;
false.
34 ?- sister(A,eric).
A = rebecca ;
false.
36 ?- brother(B,fred).
false.
39 ?- ancestor(X,eric).
X = fred ;
X = brenda ;
X = helen ;
X = john ;
X = claudeen ;
```

```
X = lloyd-sr ;
X = harley-sr ;
X = audrey ;
X = jawrence ;
X = loraine ;
X = joan ;
X = harley ;
X = elane ;
X = lloyd ;
false.
2 ?- relatives(fred,X).
X = rebecca ;
X = eric ;
X = patricia ;
X = kathy ;
X = rebecca ;
X = eric ;
X = cory ;
X = jacob ;
X = melissa ;
X = patricia ;
X = kathy ;
X = rebecca ;
X = eric ;
X = cory ;
X = jacob ;
X = melissa ;
```

```
X = harley ;
X = patricia ;
X = kathy ;
X = rebecca ;
X = eric ;
X = cory ;
X = jacob ;
X = melissa ;
X = harley ;
```

….. (Continued)

```
3 ?- decendant (brenda,X).
X = elane ;
X = lloyd ;
X = helen ;
X = john ;
X = claudeen ;
X = lloyd-sr ;
false.
4 ?- father(X,fred),brother(X,Y).
false.
```

(My dads father was an only child)

```
6 ?- aunt(Y,eric).
Y = patricia ;
Y = patricia ;
Y = kathy ;
Y = kathy ;
false.
```

```
7 ?- firstCousin(A,eric).
A = william ;
A = william ;
A = garrett ;
A = garrett ;
A = cory ;
A = cory ;
A = jacob ;
A = jacob ;
A = melissa ;
A = melissa ;
false.
8 ?- firstCousin(eric,B).
B = cory ;
B = cory ;
B = jacob ;
B = jacob ;
B = melissa ;
B = melissa ;
B = william ;
B = william ;
B = garrett ;
B = garrett ;
false.
9 ?- grandfather(A,eric), decendant (A,Y).
A = lloyd,
Y = claudeen ;
```

```
A = lloyd,
Y = lloyd-sr ;
A = harley,
Y = harley-sr ;
A = harley,
Y = Audrey;
false.
```

Problem 2 – Grammar

```
18 ?-
|    e(['(',x,+,y,')',*,'(',3,*,u,-,2,')']).
true .

19 ?- e([12, *, '(',3,/,'(',4,*,56,')',')',+,120,-,200]).
true .

20 ?- e(['(','(',2,*,x,+,5,y,+,-9,')',')']).
false.

21 ?- e([3,+,6,+,'(',5,-,6,')',8]).
false.

22 ?- e([40,*,'(',A,+,B,')']).
A = B, B = a ;
A = a,
B = b ;
A = a,
B = c ;
A = a,
B = d ;
.
.
.
.
```

(will give all options for alphabet and numbers)

```
23 ?- e(['(',12345,')']).
true ;
```

Problem 3 – Subseq sum

```
1 ?- subseqSum([ 2, 3, 5,4,6, 9 , 1] , 9, X).
X = [2, 3, 4] ;
X = [2, 6, 1] ;
X = [3, 5, 1] ;
X = [3, 6] ;
X = [5, 4] ;
X = [9] ;
false.
2 ?- subseqSum( [ 7,5,1,22,6,12,9,10,3],  20, X).
X = [7, 1, 12] ;
X = [7, 1, 9, 3] ;
X = [7, 10, 3] ;
X = [5, 6, 9] ;
X = [5, 12, 3] ;
X = [1, 6, 10, 3] ;
X = [1, 9, 10] ;
false.
3 ?- between( 10,12,M), subseqSum([3,4,5,6,7], M, Z).
M = 10,
Z = [3, 7] ;
M = 10,
Z = [4, 6] ;
M = 11,
Z = [4, 7] ;
M = 11,
Z = [5, 6] ;
```

```
M = 12,

Z = [3, 4, 5] ;

M = 12,

Z = [5, 7] ;

false.

4 ?- bagof(Z, subseqSum([3,6,5,7,8,3,5,9], 15,Z), L), length(L,N).

L = [[3, 5, 7], [3, 7, 5], [3, 3, 9], [6, 9], [5, 7, 3], [7, 8], [7, 3|...]],

N = 7.

5 ?- subseqSum( [2,3,5], A,B).

A = 10,

B = [2, 3, 5] ;

A = 5,

B = [2, 3] ;

A = 7,

B = [2, 5] ;

A = 2,

B = [2] ;

A = 8,

B = [3, 5] ;

A = 3,

B = [3] ;

A = 5,

B = [5] ;

A = 0,

B = [].
```

//In the last input, prolog is finding all possible sums (A) from the list [2,3,5] and corresponding subsequences (B) in that list [2,3,5] that will match that sum.

Problem 4 – Finite state machine

```
7 ?- fsm(0, [b,b,a,b,b,a,b]).
true .
8 ?- fsm(0, [a,b,a,b,a,b,b,a,b,b,a]).
true .
9 ?- fsm(0,[b,b,b,b,b,a,a,b]).
false.
10 ?- fsm(X, [b,b,a,a,b]).
X = 1 ;
X = 2 ;
X = 3 ;
false.
```

//The X represents all of the possible starting states that will allow the finite machine to succeed and end consuming all of the symbols.

```
11 ?- fsm(3, [a,b,a,b]).
true .
12 ?- fsm(0, [X1,X2,X3,X4]).
X1 = X4, X4 = a,
X2 = X3, X3 = b ;
X1 = a,
X2 = X3, X3 = X4, X4 = b ;
X1 = X3, X3 = X4, X4 = b,
X2 = a ;
false.
```

//X1, X2, X3, and X4, represent all of the combinations of symbols that starting from state 0, will end in a successful consumption of all symbols and end at state 3.

```
13 ?- fsm(0, [a,b,b,c,a]).
false.
```