1. (a) (9 points) Write facts and rules that define a Prolog relation \texttt{notIn(V,L)}. \texttt{V} is an atomic value and \texttt{L} is a list. The relation is satisfied (answers yes) if \texttt{V} is not a member of \texttt{L}. Thus
- \texttt{notIn(1,[])} ⇒ yes
- \texttt{notIn(1,[3,2,1])} ⇒ no
- \texttt{notIn(1,[3,2])} ⇒ yes

(b) (8 points) Explain how Prolog, using your definition of \texttt{notIn}, will solve the query \texttt{notIn(1,[2,1])}.

(c) (8 points) What will happen if, using your definition of \texttt{notIn}, we pose the following query (where \texttt{X} is a free variable)
- \texttt{notIn(X,[2,1])}.

2. (a) (15 points) What is the type of the following ML function? How did you infer the type you selected?

```ml
fun xx f [] [] = []
  | xx f [a] [b] = [f(a,b)]
  | xx f (i::t) (j::s) = f(i,j) :: (xx f t s);
```

(b) (10 points) What does the ML function \texttt{f}, defined below, compute? (\texttt{g} is used as a subroutine).

```ml
fun g [x] = []
  | g(h::t) = h::g t;

fun f [x] = x
  | f(h::t) = f(g t);
```
3. (25 points)
A well-known children’s game is “tic-tac-toe.” A three by three grid contains x’s and o’s. Three x’s is a row, horizontally, vertically, or diagonally is a winner. Assume we represent, in Prolog, a tac-tac-toe board by a list containing three sublists. Each sublist contains 3 elements, which can be an x, an o, or a b (representing a blank position). Thus the grid

\[
\begin{array}{ccc}
  x & o \\
  o & x & x \\
  o & & x \\
\end{array}
\]

would be represented as \([ [x, b, o], [o, x, x], [o, b, x] ]\).

Write Prolog rules that define the relation \(\text{winner}(\text{L})\). \text{L} is a list of lists representing a tic-tac-toe board as defined above. Given that \text{L} is ground (already bound to a value), \text{winner} should succeed if \text{L} represents a winning position for x or if \text{L} can be transformed into a winning position for x by transforming a single b into an x. That is, \text{winner} should recognize boards that x has already won or can win on his next move.

4. (25 points)
A deque is a double-ended queue; that is, a queue that allows elements to be added or removed on either end. Define an ML abstract type definition (an abstype) for a polymorphic deque. It should provide the following:

(i) \text{null}
A null deque containing no elements.

(ii) \text{empty(d)}
A boolean function that tests whether deque \text{d} is empty.

(iii) \text{enterleft(v,d)}
Return a deque with a new element \text{v} added to the left end of deque \text{d}.

(iv) \text{enterright(v,d)}
Return a deque with a new element \text{v} added to the right end of deque \text{d}.

(v) \text{rmleft(d)}
Return a deque with the leftmost element of deque \text{d} removed.

(vi) \text{rmright(d)}
Return a deque with the rightmost element of deque \text{d} removed.

(vii) \text{left(d)}
Return the leftmost element of deque \text{d}.

(viii) \text{right(d)}
Return the rightmost element of deque \text{d}.

5. (a) (18 points)
Let \text{L} be a list of distinct integers. Write a Python function \text{perm(L)} that computes a list of sublists. Each sublist is a different permutation of the values in \text{L}. For example,

\[
\text{perm([]) } \Rightarrow \text{[[[]]]} \\
\text{perm([1]) } \Rightarrow \text{[[1]]} \\
\text{perm([1,2]) } \Rightarrow \text{[[1, 2], [2, 1]]}
\]

(b) (7 points)
If list \text{L} contains duplicate values, the output of \text{perm} will contain duplicate sublists. Create a version of \text{perm}, \text{perm1}, that produces only one copy of each permutation. For example,

\[
\text{perm1([1,2,1]) } \Rightarrow \text{[[1, 2, 1], [2, 1, 1], [1, 1, 2]]}
\]
6. What do each of the following Python program fragments compute? In each case explain why.

(a) (6 points)

```
L=[3,2,1]
M=L*2
for i in L:
    M = M[1:i]+M[0:-i]
print M
```

(b) (6 points)

```
def g(a=3,b=2,c=1):
    return a+b-c
print g(c=g(), a=g(1), b=g(2,1))
```

(c) (6 points)

```
ff = map((lambda x: (lambda y:(y - x))),[1,2,3])
for f in ff:
    print f(10),
```

(d) (7 points)

```
L1=FL=range(2,25)
for i in L1:
    L2=FL;FL=()
    for j in L2:
        if i==j or j%i:
            FL=FL+(j,)
print FL
```