Instructions
Answer any four questions. (If you answer more, only the first four will count.) Each question is worth 25 points. Please try to make your answers neat and coherent. Remember, if we can’t read it, it’s wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

1. (a) (15 points)
Write Prolog facts and rules that define the relation isSublist(L1,L2). isSublist tests whether L1 is a sublist of L2. isSublist(L1,L2) is true if all of members of L1 appear somewhere in L2. Therefore L2 includes all of L1 (possibly permuted) with additional elements possibly added. For example,
isSublist([],[1,2]),
isSublist([1],[1,2]),
isSublist([2],[1,2]),
isSublist([2,1],[1,2]) and
isSublist([1,2],[1,2]) are all true, while
isSublist([1,1],[1,2]),
isSublist([2,2],[1,2]), and
isSublist([1,2,3],[1,2]) are all false.

(b) (5 points)
Explain how Prolog would solve the query isSublist([2],[1,2,3]) using your definition of isSublist.

(c) (5 points)
Would your definition of isSublist work correctly for the query isSublist(L, [1,2])? What would be returned?
2. (25 points)
Explain how type-inference works in ML. Illustrate the process by solving for the type of the following function

```ml
fun pam p f [] = []
    | pam p f (h::t) =
      if p(h)
      then f(h) :: (pam p (f o f) t)
      else (pam p (f o f) t)
```

What does this function do?

3. (a) (8 points)
Write an ML datatype definition that defines a polymorphic trinary tree, 'a triTree. A trinary tree is defined to be either a null tree, or a leaf containing a single value of type 'a, or a node that contains three subtrees of type 'a triTree. Your definition should start
```ml
datatype 'a triTree = (* and you fill in the rest *)
```
(b) (8 points)
Write an ML function count, of type 'a triTree -> int, that counts the number of leaves in a trinary tree.
(c) (9 points)
Let lessEq be a function of type ('a * 'a) -> bool that determines if one 'a value is less than or equal another 'a value.
A trinary tree is said to be ordered if its leaf values, when visited in postorder (depth-first, left-to-right order) satisfy the lessEq relation. For example, the int triTree shown below on the left is ordered because 1 ≤ 2, 2 ≤ 3, etc. However, the int triTree on the right is not ordered since 7 > 1, 5 > 3, etc.

![Ordered and Unordered Trinary Trees](image)

Write an ML function isOrdered(tree, lessEq) of type
```ml
'a triTree * ('a * 'a -> bool) -> bool
```
that takes a triTree and an ordering relation, and determines if the triTree is ordered.
4. (a) (10 points)
Write a Java class Sort that contains a member function
\[\text{void sort(int in[])}\]
that sorts an integer array, in, into sorted order. (You may use any sorting algorithm you wish).

(b) (15 points)
Now use the features of Pizza to make Sort (and sort) polymorphic. This involves two things. You must parameterize class Sort so that it can sort an array of any type. Secondly, since you must compare values to do a sort, you’ll need to add a comparison function, greaterThan, as a parameter to sort. Now sort is called as
\[\text{sort(valArray, greaterThan)}\]
valArray is an array of values to be sorted. greaterThan is a boolean function that determines whether a pair of array values are in the “greater than” relation.

5. What do each of the following Python program fragments compute? In each case explain why.

(a) (5 points)
\[L=[1,2]*3\]
\[\text{for } i \text{ in } [3,2]:\]
\[L[:i]=L[i-1:]-i\]
\[\text{print } L\]

(b) (5 points)
\[\text{def } f(a=1,b=2):\]
\[\text{return } 2*a+b\]
\[\text{print } f(b=f(f(1)),a=f(2,f()))\]

(c) (5 points)
\[\text{class T:}\]
\[a=1\]
\[\text{def m(self):}\]
\[\text{return self.a+self.b}\]
\[\text{def } \text{__init__}(self,p=2):\]
\[\text{self.b=p}\]
\[\text{print T(T.a+1).m()}\]

(d) (5 points)
\[L=[2,4,0,-1]\]
\[\text{while } L:\]
\[\text{if } L[-1]:\]
\[L=L[-1:-1]\]
\[\text{else:}\]
\[\text{break}\]
\[\text{else:}\]
\[L=[100]\]
\[\text{print } L\]
(e) (5 points)
```python
def f(x):
    return [x]*x
print map ((lambda x: [1]+f(x)),[1,2,3])
```

6. (25 points)
Consider a 3 by 3 square, containing values \( V_1 \) to \( V_9 \) as shown below.

\[
\begin{array}{ccc}
V_1 & V_2 & V_3 \\
V_4 & V_5 & V_6 \\
V_7 & V_8 & V_9 \\
\end{array}
\]

These nine values can be conveniently represented in a list of the form \([V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9]\). A square is said to be “magic” if the values in each of the three rows sum to the same value (\( V_1+V_2+V_3 = V_4+V_5+V_6 = V_7+V_8+V_9 \)) and the three columns of the square also sum to a common value (\( V_1+V_4+V_7 = V_2+V_5+V_8 = V_3+V_6+V_9 \)).

Give Prolog facts and rules that define \texttt{Magic(L)}. \( L \) is a free variable. Your solution should bind \( L \) to a the list of the integers 1 to 9, reordered so that they form a magic square as defined above.