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) (13 points)
Write a Java method

    static int[] filteredMap(int[] a, MapFct M, PredFct P)

MapFct is an object that contains a single member function declared as

    static int f(int i) { ... }

PredFct is an object that contains a single member function declared as

    static boolean p(int i) { ... }

filteredMap takes an integer array as input. P.p first filters out array elements that evaluate to false. Then M.f is applied to remaining values, which are returned as the result of the call.
For example, if M.f is defined as

    static int f(int i) {return 2*i;}

and P.p is defined as

    static boolean p(int i) {return i%2==0;}

then with a ={1,2,3,4,5,6,7,8,9,10}, filteredMap will return the array {4,8,12,16,20}.

(b) (6 points)
Rewrite filteredMap in Pizza, utilizing its first-class functions as parameters two and three.

(c) (6 points)
Extend your solution to part (b) to make filteredMap polymorphic. Now filteredMap should be able to handle any array of type T rather than just int arrays.
2. (25 points)
   Explain how type-inference works in ML. Illustrate the process by solving for the
type of the following function

   ```
   fun r(f, [a,b]) = f(a,b)
   | r(f, u::x::y::z) = f(u, r(f, x::y::z));
   ```

   What does this function do?

3. (a) (8 points)
   Write Prolog facts and rules that define the relation `last(L,E)`. L is a non-empty list
   and E is its last (rightmost) element. Your solution should work for the case that L is
   known (ground) and E is either unknown (free) or known (ground).

   (b) (17 points)
   Write Prolog facts and rules that define the relation `palindrome(L)`. L is a list of
   values (atoms or integers) that form a palindrome. A list is a palindrome if it has the
   same contents when read from left to right or right to left. That is, the list’s first ele-
   ment is the same as its last element, its second element is the same as its second from
   last element, etc.
   For example, the following lists are all palindromes: [], [a], [1,1], [a,b,a],
   [1,2,2,1],[a,b,c,b,a].
   Your solution need only work for the case in which L is ground (known).

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

   (a) (9 points)
   ```
   a = [1,2,3,4,5]
   for i in a[:-2]:
       print a[:i]+(a[-i:])*i
   ```

   (b) (8 points)
   ```
   def f(a=1,b=2,c=3):
       return a+b+c
   print f(f(),f(1,f(2)),f(2,3,f(4,5,6)))
   ```

   (c) (8 points)
   ```
   def add(v):
       return lambda li,val=v:[li]+[val]
   print map(add(10),range(1,6))
   ```
5. (25 points)
Define a Prolog relation \texttt{match(Input,Pattern)} where \texttt{Input} is a list of single character lower-case input symbols (a to z) and \texttt{Pattern} is a list of \textbf{pattern symbols}. Pattern symbols are single lower-case characters, as well as the special pattern symbols, \texttt{?} and \texttt{+}.

Pattern symbols are matched against \texttt{Input} symbols, from left to right. In a pattern, a single letter symbol matches only itself. A \texttt{?} symbol matches zero or one input characters, while a \texttt{+} in a pattern matches one or more consecutive character symbols in the \texttt{Input} list.

Your definition should answer \texttt{yes} if the pattern symbols match the input symbols, and \texttt{no} if they do not.

For example, the following queries should all answer \texttt{yes}:
\begin{verbatim}
match([a],[a]).
match([a,b],[a,b]).
match([a,b],[+]).
match([a,b],[a,a]).
\end{verbatim}

The following queries should all answer \texttt{no}:
\begin{verbatim}
match([a],[b]).
match([a,b],[a,a]).
match([a,b],[+,+]).
\end{verbatim}

Your solution needs to only handle the case where \texttt{Input} and \texttt{Pattern} are both ground (known).

Illustrate your answer by showing how the query
\begin{verbatim}
match([a,b,c],[?,a,+]).
\end{verbatim}
is solved.

6. (a) (13 points)
Let \texttt{f} and \texttt{g} be ML functions and let \texttt{L} be a list containing the values \texttt{[v 1 ,v 2 ,...v n ]}.
Write an ML function \texttt{pf(f,g,L)} that computes a list with the following \texttt{n} values:
\begin{verbatim}
[f(v 1 ,v n ),g(v 2 ,v n-1 ),...,f(v n-1 ,v 2 ),g(v n ,v 1 )]
\end{verbatim}
What is the type of \texttt{pf}?

(b) (12 points)
Let \texttt{f} be an ML function and let \texttt{L} be a list containing the values \texttt{[v 1 ,v 2 ,...v n ]}. Write an ML function \texttt{comp(f,L)} that computes a list with the following \texttt{n} values:
\begin{verbatim}
[f(v 1 ),f(f(v 2 )),f(f(f(v 3 ))),...,f^n(v n )]
\end{verbatim}
What is the type of \texttt{comp}?