Please answer the following questions, each of which is worth 20 points.

1. The question deals with object-oriented design principles.
   (a) Define the term *abstract data type*.
   (b) Draw a single class inheritance diagram for the following set of classes:
       - class Weather extends Object and adds methods low() and high().
       - class Snow extends Weather and adds methods flake() and fall().
       - class Rain extends Weather and adds instance variable “amount” and methods drip() and drop().
       - class Slush extends Snow and adds instance variable “wetness” and method splash().
       - class Sleat extends Snow and adds methods slip() and slide().

2. This question deals with algorithm analysis.
   (a) Show that the function \( f(n) = 3n^2 + 6n + 12 \) is \( O(n^2) \).
   (b) Characterize using the big-Oh notation the running time of the following algorithm:

       Let \( A \) be a given array of \( n \) positive integers.
       Let \( B \) be an array of \( n \) integers, each of which is initially 0.
       \begin{verbatim}
       for i = 0 to n - 1 do
           for j = 0 to i do
           end for
       end for
       \end{verbatim}

3. Let SimpleStack be a simple stack Java interface that only includes the two fundamental stack operations, push() and pop(), and is defined as follows:

   ```java
   interface SimpleStack {
       public void push(Object obj);
       public Object pop();
   }
   ```

   Give a short Java class that implements the SimpleStack interface using a linked list.
4. This question deals with sequences.

(a) Define what is meant by the rank of an object in a sequence.

(b) What is the output of the following code fragment of operations on a sequence, \( s \) (assume that each variable besides “\( s \)” is an Integer object representing the number it has as its name):

\[
\begin{align*}
&\text{s.insertFirst(one);} \\
&\text{s.insertLast(two);} \\
&\text{s.insertAtRank(1,three);} \\
&\text{System.out.println(((Integer) s.removeLast()).toString());} \\
&\text{s.insertAtRank(0,four);} \\
&\text{System.out.println(((Integer) s.removeAtRank(s.size()-1)).toString());} \\
&\text{System.out.println(((Integer) s.after(positionAtRank(0))).toString());}
\end{align*}
\]

5. Briefly describe an efficient algorithm to test if a string \( S \) of parentheses (i.e., “\( ( \)” and “\( ) \)” characters) is fully balanced. Recall that such a string is fully balanced if each “\( ( \)” can be matched to a unique “\( ) \)” that follows it. What is the running time of your algorithm, assuming that \( S \) has \( n \) characters?