CS383 Assignment 3

Instructions:

- *Submit all your answers in hard copies.*
- *Please submit source codes for questions 2, 3, 5 and 8 to alex_wang@126.com.*
- *This assignment is released on 11/18/2011, and due 11/23/2011.*
- *Please remember to include your name and student ID on all copies.*

1. For the listed languages, list all the basic types supported, and write a program to get the size, in bytes, of each type.
   - C++
   - JAVA
   - Pascal

2. Write a program to read in float, int, byte, double and string values, then print their bit representations. Include the results in your answer sheet.

3. Perl and Python support dynamic arrays. Implement a dynamic array class in C++. Requirements:
   a) It should be a generic array.
   b) It can contain unlimited elements.
   c) Give an analysis on the time complexity and space complexity of your implementation.

4. Define an EBNF and abstract syntax for records structures in Clite. The EBNF for structure references should use the “dot” notation discussed in the class. Modify the Clite project, the Lexer and the Parser, to make it parse a structure.

5. Upon exercise 4, write a function that judges if two structures are type equivalents. The functions should:
   a) Take two abstract syntaxes for record structure as input.
   b) Return true if they are equal, otherwise false.
   c) Three kinds of type equivalences should be considered, name equivalence, structure equivalence, structure equivalence with ignoring the fields’ order.

6. For each of the following statements using the implicit typing rules of Section 6.1, first draw the untyped abstract syntax tree, then add type information, as done in Section 6.2. If any typed abstract syntax tree is invalid, clearly indicate where and why.
   a) f = -3;
   b) i = -2.5;
   c) i = c;
   d) f = f + 1;

e) if(f1>f2) f3 = f4 + f5;

7. Show that the integer type can be defined as a recursive type starting with just the value *null*. Then write a function that checks if two integers defined in this way are equal.

8. Exercise 6.8, page 151.

9. Finish the rest of the typing rules for L{num-str} on lecture slide #36.

10. Prove by induction the following “weakening lemma” w.r.t. the L{num-str} language and its typing rules (including the ones you defined in Question 9):

[Weakening Lemma] If G |- e’ : t’, then G, x : t |- e’ : t’, for any x not in dom(G) and any type t.