308-520B Midterm 1998

This is an open book exam. You will have from 10:00 to 11:25 to complete the exam. There are 50 points in total that are distributed as follows: Q1: 8 points, Q2: 12 points, Q3: 8 points, Q4: 22 points. Please write neatly, clearly, and concisely.

1. Let us suppose that we want to implement an expression language that supports integer and string operations. Sample expressions are given below, along with the value to which they evaluate:

   ```
   3 + 4             -> 7
   4 * 5 - 2         -> 18
   aa ++ bb          -> aabb
   aa ** 3           -> aaaaaa
   a ** 4 * 2        -> aaaaaaaa
   (3 - 1) * 2       -> 4
   3 - 1 - 1         -> 1
   ```

   In short, the expression language is as you would expect for the integer operations +, -, /, and *. It is left associative, with * and / at higher precedence than + and -. We have added two new operations, ++ and **. Both of these have lower precedence than + and -. The ++ operator concatenates two strings (both the left and right operand must be strings), and the ** operator replicates a string n times (the left operand should be a string and the right operand should be an integer).

   (a) Give a non-ambiguous grammar for this language that would be suitable for a LALR(1) parser generator. Do not use directives. (5 points)

   (b) Describe the type checking rules needed to decide if an expression is correctly typed. You may use the style of type rules as given in class, or you may give clear pseudo code based on an AST representation of the expression. (3 points)

2. Assume that you have an alphabet of a, b, and c. For each of the following languages give one of the following:

   - If it is a regular language, give a regular expression that denotes the language.
   - If it is **not** a regular language, but it is a context-free language, give an unambiguous context-free grammar for the language.
   - If it is neither a regular, nor a context-free grammar, give a context-free grammar for a superset of the language, and then briefly describe a weeding routine that could be used to weed out all sentences that do not belong in the language.

   (a) All sentences that start with a, and then have any number of a's and b's. For example, a, ab, aba, $abba$. (3 points)

   (b) All sentences that have n a's, followed by n b's. For example, ab, $aabb$, $aaabbb$. (3 points)

   (c) All sentences that are palindromes (are the same string front-to-back as back-to-front). For example, a,aa,bb,aba. (3 points)

(d) All sentences that have n a's, followed by n b's, followed by n c's. For example, *abc*, *aabbcc*, *aaabbbccc*. (3 points)

3. The Java language supports arrays, where array declarations are of the form:

```
int[] data = new int[5];   /* data is an array of 5 integers */
A[]   x    = new A[10];    /* x is an array of 10 A objects  */
```

Array references are made just as in C, for example:

```
{ int sum=0;
  for(i=0; i < data.size(); ++i)
    {  sum = sum + data[i];
    }
}
```

Note that there is one extra method, `size`, that is associated with each array object.

If you were to add one-dimensional arrays to your A- JOOS compiler, briefly describe what would you need to change for:

(a) the scanner (2 points)

(b) the parser (3 points)

(c) the symbol table/type checker. (3 points)

4. Suppose that we have a language that lets us define a database as follows. Note that a database has one or more families. Each family has one or more people, and each family has its own name space. For example, one could have many families containing a person called Jean Tremblay. By default a person's name refers to the entry in the same family, although one can specifically use the `of` clause to specify a person in another family.

Each person can have up to four entries: father, mother, children and attributes. These four entries could be given in any order. Children are separated by the keyword `and`.

```
database SmithBrown

Family BrownClan {

  person Laurie Brown {
    father is Rodger Brownn;
    mother is June White;
    children are Jane Smith of SmithClan;
    attributes are [ favourite_language: "JOOS", favourite_beer: "Boreal" ]
  }

  person June White {
    children are Paul Brown and Laurie Brown;
    attributes are [ place_of_birth: "Smith Falls" ]
```

```
  }

  person Rodger Brown {
    children are Paul Brown and Laurie Brown;
    father is Alan Brown;
    mother is Elsia Green;
    attributes are [ favourite_wine: "Blue Nun" ]
  }

  person Paul Brown {
    attributes are [ favourite_sport: "Soccer" ]
    father is Rodger Brown;
    mother is June White;
    children are Doug Brown
}

Family SmithClan {
  person Jane Smith {
    father is Peter Smith;
    mother is Laurie Brown of BrownClan;
    attributes are [ favourite_food: "pickles"]
  }

  person Peter Smith {
    children are Jane Smith;
    father is John Smith;
    attributes are [ favourite_language: "Scheme" ]
  }
}
end
```

For this question:

(a) Give a grammar for this language (or a superset of this language). If you gave a grammar for a superset of the language, explain what weeding you need to do. (7 points)

(b) Give a high-level picture of what an AST for the example database would look like. You need not give all details, just enough so the overall structure is clear. (4 points)

(c) Briefly explain what your symbol table structure would look like and illustrate with a high-level picture for the example database. (4 points)

(d) Outline the sorts of type-checking or “consistency-checking” that should be done to make sure that information within each Family is consistent, and that information between families is consistent. (4 points)

(e) Based on your AST and symbol table representation, give a high-level description of a procedure, that given the family and name of a person, generates an html file listing all descendents of that person. (3 points)