University of Toronto
CSC 488S Compilers and Interpreters Winter 2007/2008

Mid Term Test (15% of course mark) March 6, 2008.

6 questions on **2** pages. **100 marks total. 50 minutes total**
Open Book and Notes, Non-programmable calculators allowed, NO other electronic aids allowed
Answer all questions. WRITE LEGIBLY!
If you need to make any additional assumptions to answer a question, be sure to state those assumptions in your test booklet.

1. [20 marks] A regular expression definition for numeric constants is given below.

Digit	(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
OptionalSign	('+' | '-' | λ)
Number	Integer | RealNumber
Integer	Digits^+
RealNumber	Digit^* ('**.**' | λ) Digit^* ((E | e) OptionalSign Digit^+ | λ)

Assume you are working with a very simple scanner that only returns single character lexical tokens.
Write a LL(1) grammar for parsing Number. Your grammar should enforce the requirement that the mantissa part of a real number must contain at least one digit. Prove that your grammar is LL(1) by showing the director sets for each rule. You may assume the set $follow(Number)$, but describe what it can not contain.

2. [20 marks] A proposal has been made to slightly simplify the project language by removing the **result** reserved word and replacing it with **return**. The two return-type statements would then be:

return
return *expression*

a)[5 marks] Does this change have any significant effect on language?
b)[15 marks] How will this change affect parsing the language? Explain your answer.

3. [10 marks] In some languages new variable *names* can be *dynamically* created at run time. For example build up a string by concatenation and then use the value of the string as the name of a new variable. This feature is heavily used to create map-like associative tables. Newly created variables have a null/zero value until assigned to.

Describe a set of runtime mechanisms that will support the creation and use of dynamically created variables.

4. [20 marks] One problem that a compiler has to deal with initialization of variables in their declaration. For example:

```
#define BIGDATASIZE   <some large integer constant>
typedef struct {
        int  count ;
        char *  name ;
        float   value ;
}  dataStruct ;
dataStruct bigData[ BIGDATASIZE ] =
    {{ 3 , "scorp" , 4.7 } , { 6 , "podger" , 9.2 } ,
     { 12, "framus" , 3.8 } , { 19 , "spanner", 7.3 } .
     /*  Assume many more values here */
}
```

a)[10 marks] Describe the semantic analysis checking required for this type of declaration.
b)[10 marks] Describe a *complete* scheme for the general case of handling storage and access to the initial value for arrays and structs with large initial values. Describe how the values are stored and how they are made available to later passes of the compiler. How hard would it be to implement this scheme in a strongly typed language like Java?

5. [15 marks] Given the C structure declaration:

```
struct L1 {
       char key ;
       double value ;
       short  kind  ;
       union {
            struct L1 * self ;
            double value2 ;
            struct {
                  char key2 ;
                  double value3 ;
            } bigVal ;
            short sData[ 9 ] ;
       } bigU ;
       struct L1 * next ;
}
```

Show how this structure would be laid out in memory using the depth first structure mapping algorithm discussed in lecture. You should assume the size and alignment factors (in bits) in the table below.

Type	Size	Alignment	Type	Size	Alignment	Type	Size	Alignment
char	8	8	int	32	32	float	32	32
short	16	16	pointer	32	32	double	64	64

6. [15 marks] Show the symbol and type table entries that a typical compiler would make for the structure declaration in Question 5.