COMP 251: 1999 Fall Semester Midterm

Date:	Wednesday, Oct 27, 1999
Time Allowed:	2 hours
Instruction:	(a) Attempt ALL questions.
	(b) Write ALL answers in the blue book provided.
	(c) The total mark is 100.

Problem 1 (10/100) Describe in English the language generated by the following grammar with <S> as the starting non-terminal:

```
<S> ::= <A> a <B> b <C> c
<A> ::= a <A> | a
<B> ::= <B> b | b
<C> ::= c <C> | c
```

Problem 2 (18/100) Given the following grammar for expressions:

```
 <E> ::= - <E1>
<E1> ::= <E1> + <E2> | <E2> * <E1> | <E2>
<E2> ::= <E2> / <E3> | <E3>
<E3> ::= a | b | c
```

(a) Describe the associativity rule for the operators `+, *, /, -`.

(b) Give a precedence table about the four operators.

(c) Modify the grammar to add the operator "$\wedge$" which is right-associative and has the highest precedence.

Problem 3 (16/100) Give a procedure example to show the difference in parameter passing between

(a) call-by-value and call-by-reference.

(b) call-by-value and call-by-value-result.

(c) call-by-reference and call-by-value-result.

Note: Only an example is required in each case; and, no explanation is needed.

Problem 4 (16/100) Given the following C-like program, draw the flow of activation records in the style of Assignment #2 when the main() is executed .

```
int i = 1;

int greater(int x, int y) { if (x > y) return x else return y; }

void main()
{
    int i = 5, x = 3;
    x = greater(i,x);
}
```

Problem 5 (16/100) What is the type of F in each of the following SML functions? Use notations of the SML interpreter in our labs.

(a) `fun F x = if x > 0 then ''good'' else ''bad''`

(b) `fun F [] = [] | F((a,b)::tail) = (a+b)::(F tail)`

(c) `fun F x y = x y`

(d) `fun F(x,y,z) = map x (map y z)`

Problem 6 (24/100) SML Programming.

An integer arithmetic expression with only additions is represented by a list of string tokens, one token for each operand and each "+" operator. e.g. the expression, $12 + 5 + 33$ is represented as: `[''12'', ''+'', ''5'', ''+'', ''33'']`.

Assume that you are given an SML function str2int which converts a string to an integer if the input is a string representation of the integer (similar to atoi() in C). e.g.

str2int "256";

will return 256. On the other hand, if the input is not an integer, it will raise the exception NOT_INT. Finally str2int "" returns 0.

(a) Write a polymorphic SML function odd_list to check if a list contains an odd number of items. You are **NOT** allowed to use the build-in list function "length".

`Example` : odd_list ["a", "b", "c", "d", "hkust"]

`Example Output` : true

(b) Write an SML function odd_item to get the first, third, fifth, ... items from a string list. At the same time, you have to check that the even items are all "+"; if not, raise the exception WRONG_FORMAT.

`Example` : odd_item ["a", "+", "c", "+", "hkust"]

`Example Output` : ["a", "c", "hkust"]

(c) Write an SML function strl2intl to convert a list of integer strings to an integer list by using the given function str2int. Any string item that does not represent an integer should be converted to zero by specifying an appropriate handler for the exception NOT_INT.

`Example` : strl2intl ["1", "hkust", "3"]

`Example Output` : [1, 0, 3]

(d) Write an SML function sum_int to add a list of integers.

`Example` : sum_int [1, 2, 3]

`Example Output` : 6

(e) Write an SML function eval_iexpr to evaluate an integer arithmetic expression as described with the following requirements:

- You may use any of the functions you define above in part (a) - (d). If you need any other auxiliary functions, they must be defined **locally** using the let statement.
- Raise the exception WRONG_FORMAT if the length of the input list is not odd or the even items are not "+".
- In SML, the following constructs deal with exceptions:
  - exception
  - raise
  - handle =>

`Example` : eval_iexpr ["12", "+", "mak", "+", "4"]

`Example Output` : 16