HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY

COMP3031 (Principles of Programming Languages)

Fall 2011

FINAL EXAMINATION

Dec 21, 2011 (Wed) 8:30am-11:30am, LG1 Table Tennis Room

Name **SOLUTION**	**ITSC Account**
Student ID	**Lecture Section** L1

1. *<u>About the exam:</u>*
   a. *This is a closed-book, closed-notes examination.*
   b. *You CANNOT use any electronic device (e.g. calculator) during the examination. Please TURN OFF all of your electronic devices (e.g. mobile phone) and put them in your bag.*
   c. *You cannot leave during the last 15 minutes of the examination.*
2. *<u>How to answer:</u>*
   a. *You can ONLY use ball pen with black or blue ink to write your answers. Your answer should be in the designated space following each question.*
   b. *Rough work can be done in the provided "additional blank paper for draft work". However, do not write answers there as they will NOT be graded.*
3. *<u>About this paper:</u>*
   a. *There are a total of 11 pages, including this page.*
   b. *There are a total of 8 questions counting for 100 points in total.*
   c. *Attempt all questions. Partial credit will be given as applicable.*

Problem	1	2	3	4	5	6	7	8	Total
Points									

Problem 1 (10 points) SML Programming

A point is represented as a 2-integer tuple (X,Y) of X and Y coordinates. A point cloud is represented as a list of points. A point cloud is *clear* if there are no identical points in the list. A point cloud is *sorted* if all points in the list are first sorted on X and then on Y, both in ascending order. By definition, an empty cloud is both clear and sorted.

(a) Write a function *clear*: (int * int) list -> bool to determine whether a point cloud is clear.

```
fun in_cloud((x:int,y:int), []) = false |
      in_cloud((x,y), (x1,y1)::tail) =
         if x=x1 andalso y=y1 then true else in_cloud((x,y), tail);

fun clear([]) = true |
      clear(head::tail) = if in_cloud(head,tail) then false else clear(tail);
```

Grading criteria

1. Compare two 2-integer tuple [X,Y] (1 point)
2. Determine whether a 2-integer tuple [X,Y] appeared in a point list L (2 points)
3. Determine whether a point cloud is clear. (2 points)

(b) Write a function *sort*: (int * int) list -> (int * int) list such that the output point cloud is sorted from the input point cloud.

Grading criteria:

1. Compare relative value of two 2-intger tuple [X1,Y1] and [X2, Y2]. (1 point)
2. Bubble sort/ Insertion sort/ Quick sort/other sort (4 points)

```
fun dcomp ((x1:int,y1:int), (x2:int,y2:int))
= if x1<x2 then true else if x1=x2 then (y1<y2) else false;

fun insert ((x1,y1), []) = [(x1,y1)] |
      insert ((x1,y1), (x2,y2)::t) = if dcomp((x1,y1),(x2,y2)) then (x1,y1)::(x2,y2)::t
                                     else (x2,y2)::(insert((x1,y1),t));
fun sort [] = [] |
      sort ((x1,y1)::t) = insert((x1,y1), sort t);
```

Problem 2 (10 points) Prolog Programming

A point is represented as a 2-integer list [X,Y] of X and Y coordinates. A point cloud is represented as a list of points. A point cloud is *clear* if there are no identical points in the list. A point cloud is *sorted* if all points in the list are first sorted on X and then on Y, both in ascending order. By definition, an empty cloud is both clear and sorted.

(a) Write a predicate *clear(P)* that returns *true* if the given point cloud *P* is clear.

```
clear([]).                                  (1 point)
clear([X|Y]) :- \+ member(X,Y), clear(Y).   (2 points)
```

(b) Write a predicate *psort(P1, P2)* such that the point cloud *P2* is sorted from the given point cloud *P1*.

```
sorted([X1,Y1],[X2,Y2]) :- X1<X2.                  1 point
sorted([X1,Y1],[X2,Y2]) :- X1=X2, Y1<Y2.           1 point

psort([],[]).                                      1 point
psort([H|T],S) :- psort(T,St), insert(H,St,S).     1 point

insert(H, [], [H]).                                1 point
insert(H, [H1|T1], [H, H1|T1]) :- sorted(H,H1), !. 1 point
insert(H, [H1|T1], [H1|S]) :- insert(H,T1,S), !.   1 point
```

Problem 3 (12 points) Grammars and Regular Expressions

(a) Write a regular expression to represent a language L consisting of all strings of characters a's and b's that do **not** contain any substring "aaa". For example, "aa", "b", and "aba" belong to L whereas "baaa" does not.

(b|ab|aab)*a?a? (4 points)

(b) Write an **unambiguous** context-free grammar to describe a type definition <S> satisfying the following specifications:
(i) A basic type "int", "char", and "float" is a type;
(ii) An array type, which is a pair of brackets "[]" following a type, is a type;
(iii) A pointer type, which is a star "*" followed by a type, is a type.
(iv) "[]" has a higher precedence than "*".

<S> ::= *<S> | <T>
<T> ::= <T>[] | <B>
<B> ::= int | char | float

Grading criteria:

1. Give basic type int char float. (2 points)
2. Give right array type. (2 points)
3. Give right pointer type. (2 points)
4. Ensure [] has a higher precedence than *. (2 points).

Problem 4 (15 points) Flex and Bison

Given the following grammar:

<Expr> ::= <Expr> <Term> | <Term>
<Term> ::= <Unit> <Op> | <Unit>
<Unit> ::= (<Expr>) | <Id>
<Op> ::= * | ? | +
<Id> ::= a | b | ... | z

The Flex file for strings specified by this grammar is also given:

```
%option noyywrap

%{
#define YYSTYPE int
#include "regex.tab.h"
%}

op  [()*?+]
ws  [ \t]+

%%
[a-z]   return CHAR;
{op}    |
\n      return *yytext;
{ws}    /* eat up white spaces */
%%
```

Complete the following BISON file to count the total number of occurrences of the three wildcards *, ?, and + in a string that satisfies the grammar:

```
%{
#define YYSTYPE int
#include <stdio.h>
%}


%token CHAR


%%
input: /* empty */
       | input line
       ;

line: '\n'
       | expr '\n' { printf("\t%d\n", $1); }
       ;

/* ADD BISON PRODUCTION RULES AND ACTIONS HERE FOR THE GIVEN CFG */

expr: expr term    {$$ = $1 + $2;}
       | term      {$$ = $1;}
        ;

term: unit '?'     {$$ = $1+1;}
       | unit '*'  {$$ = $1+1;}
       | unit '+'  {$$ = $1+1;}
       | unit      {$$ = $1;}
        ;

unit: '(' expr ')' { $$ = $2; }
        | CHAR { $$ = 0; }
        ;

%%


int main()
{
    return yyparse();
}


int yyerror(const char* s)
{
    printf("%s\n", s);
    return 0;
}
```

Grading criteria:

1. On the basis of correct grammar:
    a. 1 action sentence. (1 point).
    b. Proper quotation marks and semicolon. (1 mark each).
2. Incorrect grammar:
    a. Below 7 point. Depend on the case.

Problem 5 (10 points) Prolog Programming

Define a predicate *join(X,Y, Z)* in Prolog: Given two integer lists *X* and *Y*, return in *Z* a list of all 2-integer lists where each 2-integer list contains a pair of identical integers from X and Y. The order of the 2-integer lists in Z does not matter. Assume X and Y are given, there is no duplicate element in X, and there is no duplicate element in Y.

For example:

?- join([], [1,2], Z).
Z = [].

?- join([1,2], [1,3], Z).
Z = [[1,1]].

?- join([1,2], [1,2,3], Z).
Z = [[1,1], [2, 2]].

```
join([],_,[]).                                          2 points
join([H|T],L, [[H,H]|J]) :- member(H,L), !, join(T,L,J).   4 points
join([H|T],L,J) :- join(T,L,J).                         4 points
```

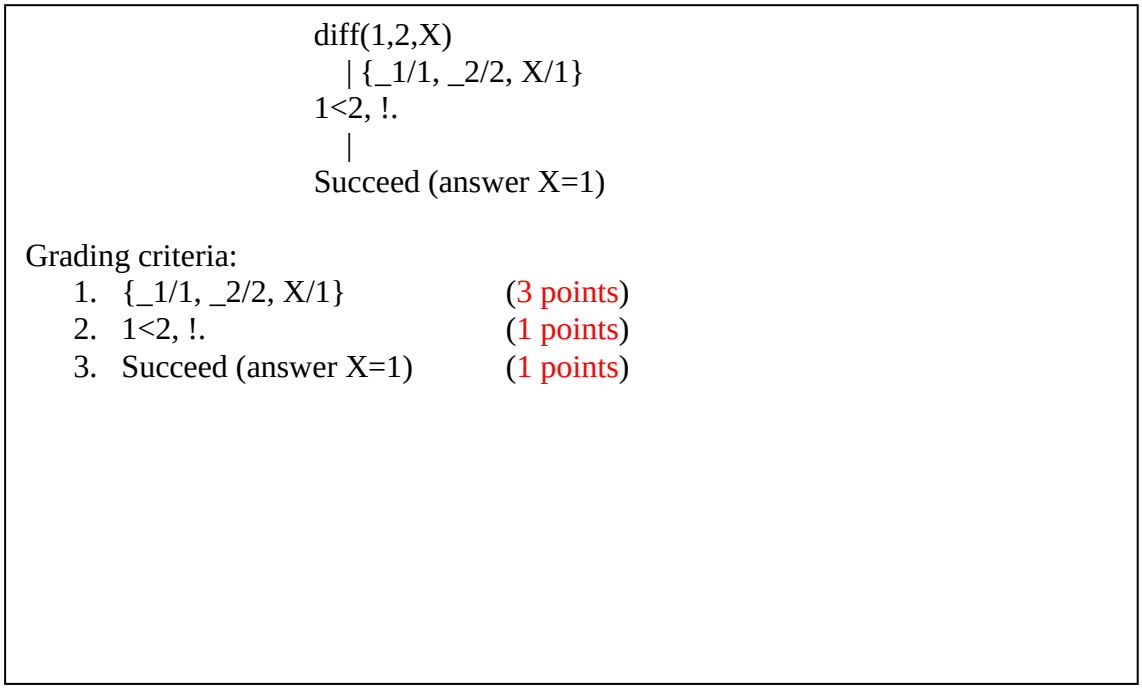
Problem 6 (10 points) Prolog Search Tree

Given the following Prolog program:

diff(X,Y,1) :- X < Y, !.
diff(X,Y,1) :- X > Y, !.
diff(X,X,0).

Draw the complete Prolog search tree for each of the following two queries:

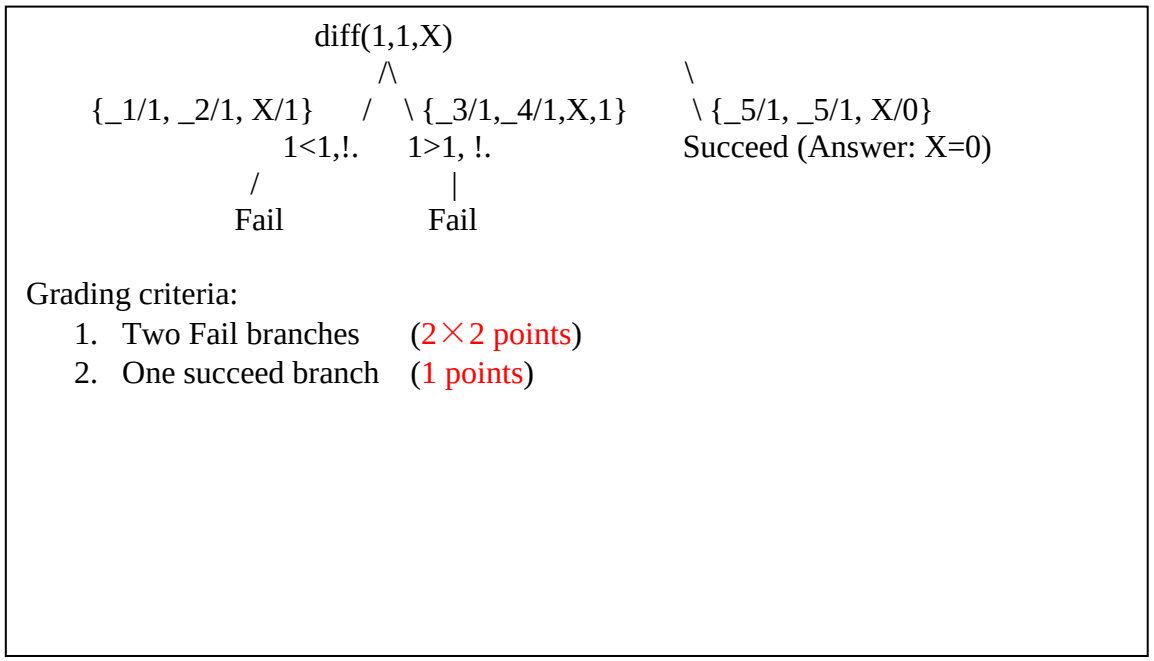
(a) *diff(1,2,X).*



Grading criteria:

1. {_1/1, _2/2, X/1} (3 points)
2. 1<2, !. (1 points)
3. Succeed (answer X=1) (1 points)

(b) *diff(1,1,X).*



Grading criteria:

1. Two Fail branches (2×2 points)
2. One succeed branch (1 points)

Problem 7 (18 points) Parameter Passing Methods

Suppose an imaginary D language has syntax similar to the C language, but can apply a parameter passing method as specified. Determine the output of the following D program with each specified parameter passing method:

```
int count=1;
void forward(int steps) {
   steps++;
   printf("%d ", count);
   count = count+steps;
   printf("%d ", count);
}
int main(int argc, char **argv) {
   count++;
   forward(count);
   printf("%d ", count);
}
```

Output:

With Call-by-Value:

2 5 5

With Call-by-Reference:

3 6 6

With Call-by-Value-Result:

2 5 3

Grading criteria:

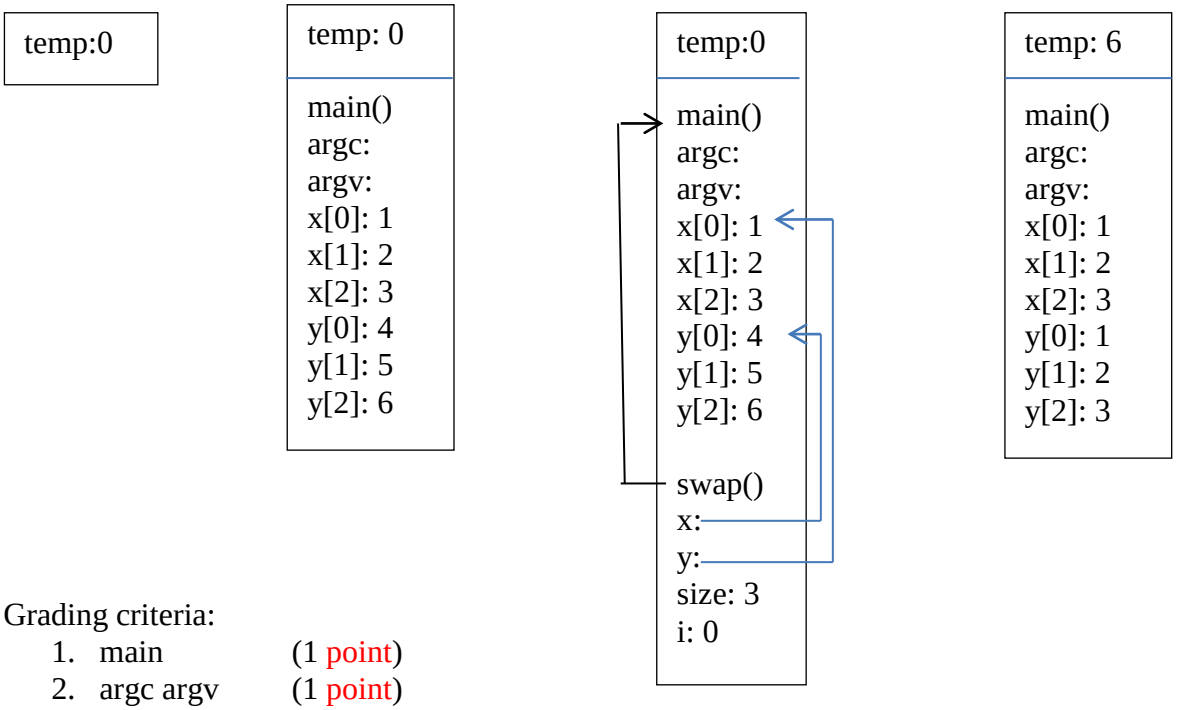
2 points for each integer value

Problem 8 (15 points) Activation Records

Recall that the C language by default uses *static scoping* on variable names and *passing-by-value* for parameter passing in procedure calls. Complete the activation records, including the variables and their values if known, the parameters and their values if known, the procedure calls, and the control links, for the following C program:

```
#include <stdio.h>
int temp=0;
void swap(int x[], int y[], int size) {
   int i=0;
   while (i<size) {
      temp = x[i];      x[i] = y[i];      y[i] = x[i];
      i++;
   }
}
int main(int argc, char **argv) {
   int x[3] = {1,2,3},
       y[3] = {4,5,6};
   swap(y,x,3);
}
```

Activation Records:



Grading criteria:

1. main (1 point)
2. argc argv (1 point)
3. three temp (3×1 points)
4. x and y array in first and second form (2×1 points)
5. x and y array in third form (2×1 points)
6. one control link(black line) (1 point)
7. two pointers(blue line) (2×1 points)
8. swap (1 point)
9. size (1 point)
10. i (1 point)