American University of Beirut

CMPS 258

Programming Languages

Spring 2004

Final Exam

Date: May 31st, 8:00 – 10:00am. **Name: ----------**

Instructor: Chiraz Ben Abdelkader

- This exam is closed-book and closed notes. You are allowed to use **two A4 pages of notes**, front and back.
- There are a total of **7 pages** and **6 problems** for a total **100 points**.
- Please write as legibly as you can, otherwise I will NOT mark your answer!

	Prob. 1	Prob. 2	Prob. 3	Prob. 4	Prob. 5	Prob. 6	Total
Max Grade	25	12	16	22	10	15	**100**
Your Grade							

Question 1. (25 pts) Short-answer questions, 5 pts each

a) Consider the following function definition in Scheme:

```
(define (foo x)
    (IF (null? x) '())
    (IF (list? x) (car x))
    (IF (or (number? x) (symbol? x)) x '())
)
```

Based on this definition, determine the value of each of the following s-expressions:

```
(foo 10)                  -->
(foo '((A (B)) C D))      -->
(foo 'A)                  -->
```

b) Explain why in Prolog subgoals generally must not be proven independently (or separately) of each other; illustrate with an example. Also give an example of a compound goal statement (i.e. one containing multiple subgoals) for which it is actually okay to prove the subgoals independently.

c) Explain what it means for Prolog to use a depth-first, left-to-right proof strategy.

d) Explain why class constructors and destructors in C++ can not have a return value.

e) Give the meaning in English of each of the following Prolog statements. Be sure to specify the corresponding quantifiers.

- Rule: `sibling(X,Y) :- mother(Z,X), mother(Z,Y).`
- Goal: `mother(_,X).`
- Goal: `aunt(_,_).`

ANSWERS (use back side if necessary)

Question 2. (12 pts) Variables

Consider following C code:

```
class S1 {public: S1(); private: int count; /* ... */ };
int N=10;
S1 *a[10];
float x;
void main()
{                                                    <-- Point1
      char *str;
      str = malloc(20);
      x = 4;                                         <-- Point2
      first(N);
      free(str);
}
void first(int z) {
      S1 *a2[10];
      {
         int i;
         for (i=0; i<z; i++)
             a[i] = malloc(2);
      }
      z = z+1;                                       <-- Point3
}
```

a) List the kind of storage binding (i.e. static, stack-dynamic, heap-dynamic) to the right of each variable declaration or allocation.

b) What is the referencing environment at Point1, Point2 and Point 3?

c) Which variables are bound to storage when program execution reaches Point1, Point2 and Point3?

ANSWERS (use back side if necessary)

Question 3. (16 pts) Consider the following C++ code,

```
class A {
public:
     void f1();
     virtual void f2();
     void f3();
};

class B : public A {
public:
     void f2();
     void f3();
private:
     int count;
};

class C: public B {
public:
     void f1(float);
     void f2();
};

class D: {
friend class B;
public:
     void f1();
     void f2(int, float);
};

void main() {
   A a, *pA;
   B b;
   C c;
   D d;
}
```

a) For each of the objects a, b, c and d in main, indicate which member methods can be called/accessed by that object. Use the scope operator to indicate the class of each method; for example A::f1(). Also indicate when a method is **overridden** or **overloaded**.

b) For each of the objects a, b, and c, indicate which member methods can be called if the object were referenced through pointer pA instead.

ANSWERS (use back side if necessary)

Question 4. (22 pts) Prolog

a) Consider the following fact and rule statements in Prolog:

```
sibling(X,Y) :- parent(Z,X),parent(Z,Y)
parent(mary,john).
parent(mary,mike).
```

Show how the resolution logical inference rule can be applied over multiple (two) steps to show that `sibling(mike,john)` is true.
Hint: you will need to make appropriate variable substitutions. Clearly indicate any such substitutions made in each resolution step.

b) Consider the following fact and rule statements in Prolog:

```
may_steal(Person,Thing) :- thief(Person),
                           likes(Person,Thing), valuable(Thing).
likes(joe,fish).
likes(joe,mary).
likes(mary,fish).
likes(mary,joe).
thief(joe).
valuable(X).
```

Based only on these rules and facts, what would be Prolog's answer to the query: `may_steal(X,Y)`. If the answer is *yes*, give all the possible values of X and Y.

c) Write Prolog rules to express the following human relationships:
`Relative(X,Y)` : A person is your relative if s/he is your mother, father, grandmother, grandfather, sibling, aunt, uncle, nephew, niece, first cousin, or second cousin.
`CannotMarry(X,Y)` : You cannot marry someone who is your relative or who is of the same gender.
`Aunt(X,Y)` : Your aunt is your parent's sister or the wife of your parent's brother.

d) Consider the following rule in Prolog:

```
CanMarry(X,Y) :- not(CannotMarry(X,Y)).
```

Suppose Prolog answers the query `CanMarry(john,ann)` with a *yes*. Does this really imply that John and Ann indeed *can* marry? Discuss how this relates to the closed-world and negation limitations of Prolog?

ANSWERS (use back side if necessary)

Question 5. (10 points) Pointers in C

Consider the following C code:

```
void foo(int *left, int *right) {
   int temp = *right;
   *left = *right * 2;
   *right = temp / 2;
}

void main() {
    int A = {15,0,21,-11,3};
    int *p1, *p2;
    p1 = &A[0];
    p2 = &A[4];
    int i = 0;
    for (i=1; i<=5; i++) {
       foo(p1,p2);
       p1++;
       p2--;
    }
}
```

Assuming the base address of array A is 3120, fill in the table below showing the **memory contents** of each variable right before the loop (i=0) and at the end of each **for** loop iteration (i=1,2,3,4,5).

	Iterations (i)					
Variables	0	1	2	3	4	5
A	15,0,21,-11,3					
p1	3120					
p2						

Question 6. (15 pts) Scheme

a) Suppose you have a list of pairs in which symbols are paired with an integer. For e.g.

```
((Mark 15) (Angela 27) (Joe 11))
```

Write a Scheme function **update** that takes a list `lis` of the above form and a symbol `symb`. If the symbol is already in the list, then it returns a list with the corresponding count updated. If the symbol is not present in the list, then it adds the pair (`symb` 1) to the list. So for example:

```
(update 'Joe '((Mark 15) (Angela 27) (Joe 11)))  -->
((Mark 15) (Angela 27) (Joe 12))

(update 'Nadia '((Mark 15) (Angela 27) (Joe 11))) -->
((Mark 15) (Angela 27) (Joe 11) (Nadia 1))

(DEFINE (update symb lis)
```

b) Write a Scheme function **set-equal?** that tests whether two given sets S1 and S2 are equal. A set is represented as a list and its members are the top-level elements of that list. Two sets are equal if they contain exactly the same members, ignoring ordering. For example,

```
(set-equal? '(1 2 3) '(3 2 1)) --> #t
(set-equal? '(A B) '((A) B)) --> ()
```

You may assume that S1 and S2 do not contain duplicate members.

```
(DEFINE (set-equal? S1 S2)
```

c) Write a Scheme function **set-intersect** that computes the intersection of two given sets S1 and S2, as defined in (c). For example,

```
(set-intersect '(3 2 1) '(1 4)) --> (1)
(set-intersect '(A B) '((A) B)) --> (B)
```

Again, you may assume that S1 and S2 do not contain duplicate members.

```
(DEFINE (set-intersect S1 S2)
```

ANSWERS (use back side if necessary)