METU, Department of Computer Engineering
CENG 242 - PROGRAMMING LANGUAGES CONCEPTS
FINAL EXAM (Spring 2009)
CLOSED NOTES AND BOOKS,5 questions, 6 pages, 100 points, DURATION: 120 mins

NAME:_____ **ID:**_____

QUESTION 1. (20 pts)

Answer the following short questions in the boxes:

a) $(\lambda x.\lambda y.x + (\lambda x.x + 1)\ \ (x + y))\ \ (\lambda z.z - 4\ \ 5)\ \ 10 =$ 13

b) Haskell expression:

```
let a = 1
    f x = a+g x
    g x = x+2
in f 2 + let a = 4
             g x=x+1
         in f 3 =
```
11

c) Haskell expression assuming dynamic scope/binding:

```
let a = 1
    f x = a+g x
    g x = x+2
in f 2 + let a = 4
             g x=x+1
         in f 3 =
```
13

d)

```
data Egg = Eaten | Born Chicken
data Chicken = Male | Female [Egg]
```

Assuming you have these data types, write a value of a chicken that's mother of a rooster (male chicken), mother of a chicken with no eggs, and having two eggs eaten.

```
Female [Born Male, Born (Female []), Eaten, Eaten]
```

e) In C, assuming `int a,b; double x,y; x = a/b; y = x/y;`
Division is either integer division or floating point division. What kind of overloading C uses for division?

context independent overloading

QUESTION 2. (20 pts)

Assume you are asked to write a polymorphic *list* data structure that needs objects to be stored in a specific order defined by the class of the contained objects. Assume all contained class objects need to define two member functions:

```
int compare(const Object &) const;
void show() const;
```

compare() returns negative for less than, 0 for equality and positive for greater than relation(i.e. a.compare(b) returns -1 if $a < b$). show() outputs the element on standard output. Your data structure class TheList should implement member functions:

```
void insert(Object *);
void show();
```

for inserting a new element to the list and outputting all elements in the list respectively. Elements are inserted in increasing order based on the compare() method of the objects.

a) Provide the definition of TheList data structure using C++ templates (generic abstractions). Only provide the member variables, insert () and show() functions of the class. In function bodies, only indicate how you use object methods compare() and show(). **Do not give full function definitions, just calls you make**.

b) Assume you want to reach polymorphism through late binding (using abstract classes and virtual members with inheritance). Provide TheList structure with similar information with the first part. In addition provide the abstract class definition, and a sample contained class definition having only one integer as the member variable.

c) In 3 sentences, tell the advantage/disadvantage of templates versus polymorphism through late binding in such an application.

a)
```
template<class Object>
  class TheList {
        struct Node {
                Object *content;
                Node *next;
        } *root;
  public:...
        void insert(Object *a) {
                Node *p;
                ....
                if (a->compare(*(p->content))) ...
                ..
        }
        void show() {
                Node *p;
                ...
                p->content->show();
                ...
        }
};
```

b)
```
class Object {
        virtual int compare(const Object &) const;
        virtual void show() const;
};
class TheInt {
        int x;
public:
        virtual int compare(const Object &a) const {
                TheInt *ia=(TheInt *)&a;
                return ia.x-x;
        }
        virtual void show() const {
                cout << x ;
        }
};
class TheList {
        struct Node {
                Object *content;
                Node *next;
        } *root;
  public:...
        void insert(Object *a) {
                Node *p;
                a->compare(*(p->content));
        }
        void show() {
                Node *p;
                ...
                p->content->show();
                ...
        }
};
```

QUESTION 3. (20 pts)

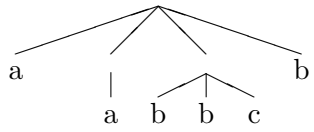
```
class A { protected:
        int a;
public: A(int ap) { a=ap;  cout << "A(" << ap << ")\n";}
};
class B:public A {
        int a,b;
public: B(int ap, int bp):A(ap),a(ap)
                { b=bp; cout << "B(" << b << ")\n";}
        B(const B & bo):A(bo.A::a)
                { b=bo.b; a=bo.a ; cout << "B(COPY)\n";}
        ~B()    { cout << "~B(" << a << ',' << b << ")\n";}
        // decrement all members and return if any of them is zero
        int decifanyzero()  { a--; b--; A::a--; return !(a && b && A::a);}
        // test if all members are zero or negative
        int allnonpositive() { return a<=0 && b<=0 && A::a<=0;}
};
int f(B n) {
        int t;
        t=n.decifanyzero();
        if (t) throw 0;
        if (n.allnonpositive())
                return 1;
        else
                return f(n);
}
int main() {
        B a(2,4);
        try {    f(a);
                cout << "successful\n";
        } catch (int a) {
                cout << "exiting\n";
        }
}
```

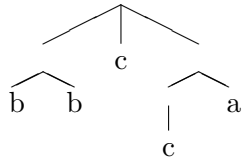
What is the output of the C++ program above. (Reminder: `throw` destructs all local variables in intermediate activation records properly)

```
A(2)
B(4)
A(2)
B(COPY)
A(1)
B(COPY)
~B(0,2)
~B(1,3)
exitting
~B(2,4)
```

QUESTION 4. (20 pts)

a) A Bare Syntax Tree (BrST) is a tree in which nodes can have variable number of subtrees, and only leaves have data, say `a,b,c`. Some BrSTs are shown below.





Design a Prolog data structure for BrST, to represent BrSTs as instances of a `brst` predicate. Note that BrSTs are trees that may contain a list of BrSTs. But they are not lists themselves because for example `[b,c,c]` is not a BrST but `([b,c,c])` is a BrST.

Write both trees above as instances of the `brst` predicate you designed for the representation.

tree is node([list of tree or leaf])
leaf is atom

```
node([a,node([a]),node([b,b,c]),b])
node([node([b,b]),c,node([node([c]),a])])
```

b) Consider the following Prolog code.

```
orbits(mercury,sun).
orbits(earth,sun).
orbits(moon,earth).
orbits(europa,jupiter).

planet(B) :- orbits(B,sun).

satellite(B) :- orbits(B,P), planet(P).
```

```
satellite(X).

X = moon ? ;

no
```

How can we ask Prolog to give us all the satellites, and what is the answer to this question according to the program above?

QUESTION 5. (20 pts)

Suppose that you are going to design a new programming language. You are expected to implement two binary operations and one unary operation below.

$M \circ N$	left-associative	1
$M \bowtie N$	right-associative	2
$\star M$	non-associative	3

M and N can be single-letter identifiers, say a, b, c, d. For example, $a \circ b \bowtie (c \circ d)$ and $\star(a \circ b)$ are fine, but $a \bowtie \star \star d$ and $ab \bowtie$ are not.

Suppose also that you are going to implement the precedence and associativity of these operators in a grammar (i.e. the parser will rely on this grammar to understand their order of execution). Their precedence and associativity are given above, where higher number means higher priority. Parentheses override all priorities (i.e. a parenthesized expression has highest priority.)

a) Write a fragment of grammar to faithfully describe the precedence and associativity of these operators.

$$
\begin{array}{lcl}
O & :: & O \circ C \;\mid\; C \\
C & :: & L \bowtie C \;\mid\; L \\
L & :: & \star Lns \;\mid\; T \;\mid\; (O) \\
T & :: & a \;\mid\; b \;\mid\; c \;\mid\; d \\
Ons & :: & O \circ C \;\mid\; Cns \\
Cns & :: & L \bowtie C \;\mid\; Lns \\
Lns & :: & T \;\mid\; (Ons)
\end{array}
$$

b) Is your grammar top-down parsable? Briefly explain why (not).

No. It is left recursive. O is expanded to O infinitely when trying the rule at line 1.