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

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)

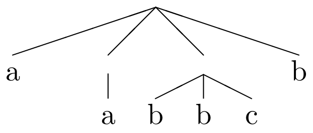
```
class A { protected:
        int a;
public: A(int ap) { a=ap;  cout << "A(" << ap << ")\n";}
};
class B:public A {
        int a,b;
public: B(int ap, int bp):A(ap),a(ap)
                { b=bp; cout << "B(" << b << ")\n";}
        B(const B & bo):A(bo.A::a)
                { b=bo.b; a=bo.a ; cout << "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";
        }
}
```

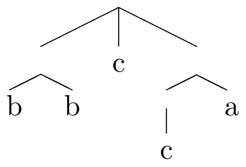
What is the output of the C++ program above. (Reminder: `throw` destructs all local variables in intermediate activation records properly)

```
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.

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

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.

Name, SURNAME and ID ⇒

Middle East Technical University

Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring '2014-2015

Final Exam

- **Duration: 120** minutes.
- **Total Points: 100**
- **Exam:**
  - This is a **closed book**, **closed notes** exam. The use of any reference material is strictly forbidden.
  - No attempts of cheating will be tolerated.
- **This exam consists of 10 pages including this page. Check that you have them all!**
- **GOOD LUCK !**

Question 1

Question 2

Question 3

Question 4

Question 5

Question 6

Total ⇒

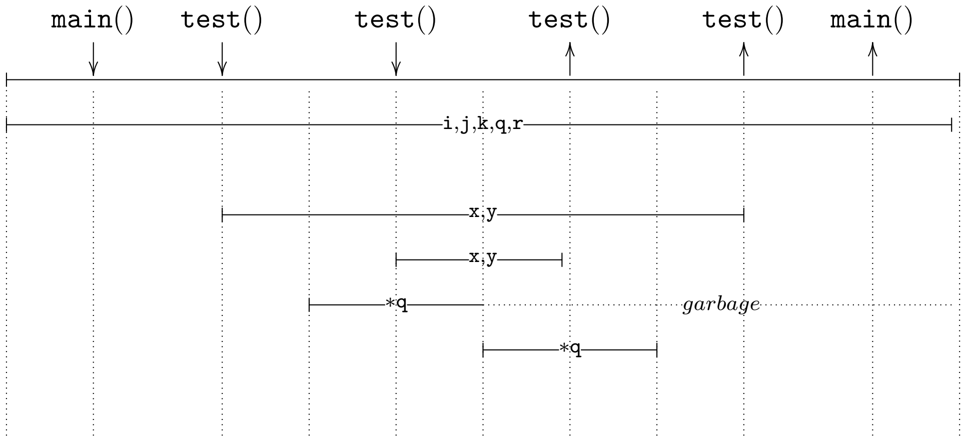
QUESTION 1. (15 points)

In the following C program (assume C features only) determine garbage variable(s) and dangling reference(s). Explain how and where they have occurred. You need to trace the execution of the program and keep track of the lifetimes and the contents of all the variables in order to discover garbage variables and dangling references. So, do the followings:

- Show the lifetimes of all variables on lifetime chart below (add necessary points to the chart for creating/destroying heap variables)
- Show how/if dangling reference and garbage variable occurs on lifetime chart below (such as reference time)
- Determine the output

```
#include "stdio.h"
#include "malloc.h"

int i=0, j=1, k=2;
int *q, *r;
void test(int x, int y)
{
     q=(int *)malloc(sizeof(int));
     *q=y;
     r=&k;
     x++;
     y++;
     (*q)++;
     (*r)++;
     if (x<2) {test(x,y); free (q);}
}
main()
{
     test(i,j);
     printf("%d %d %d %d %d\n",i,j,k,*q,*r);
}
```



`0 1 4 *q 4`
(`*q` is dangling reference)

QUESTION 2. (20 points)

a) (10 pts) Determine the output of the following program (written in a C like language) assuming static binding for the following parameter passing mechanisms:

a) normal order evaluation (call by name)
b) definitional mechanism, variable (call by reference)

```
int a[3]={10,20,30};
int i=1;
void test(int x, int y, int z)
{
      x++;  y--;  z++;
      printf("%d\n",z);
      x--;  y++;  z--;
      printf("%d\n",z);
      a[0]++;  a[1]--;  a[2]++;
      printf("%d %d %d \n",a[0],a[1],a[2]);
}
main()
{
      test(i,a[0],a[i]);
}
```

a) OUTPUT - by name

31

19

11 18 32

b) OUTPUT - by reference

21

20

11 19 31

b) (10pts) Determine the **output** of the following program (written in a C like language) assuming dynamic binding and call by value parameter passing technique is used. Determine **the environments** at the start time of each function. For each identifier specify where it is declared (such as `a->global int, a->main int`, etc.).

```
int i=5, j=5;
void g(int k)
{                   //E(g) = {i->main, j->f, k->g, f , g, main      }
        k=i+j+k;
        printf(   %d    ,k);
}
void f (int j)
{                   //E(f) = {i->main, j->f, f , g, main      }
        j=i+j;
        g(j);
        printf (   %d    ,j);
}

void main()
{                   // E(main) = {i->main, j->global, f , g, main      }
    int i=10; f(i); printf(   %d   , i);
}
```

OUTPUT

50

20

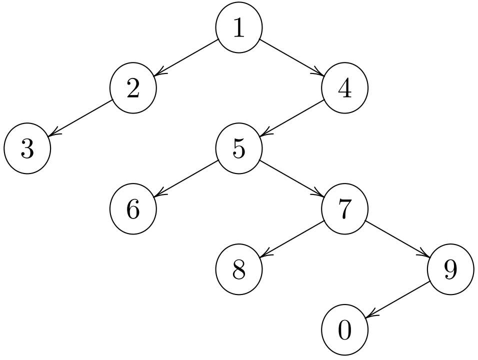
10

QUESTION 3. (20 points)

a) (10 points) Consider the following data type definition used for generating trees:

```
data TREE = EMPTY | NODE (Int, TREE, TREE)
```

Examples corresponding to the following tree are as below:



```
mytree = NODE (1, NODE (2, NODE (3, EMPTY, EMPTY),
                           EMPTY),
                  NODE (4, NODE (5, NODE (6, EMPTY, EMPTY),
                                    NODE (7, NODE (8, EMPTY, EMPTY),
                                             NODE (9, NODE (0, EMPTY, EMPTY),
                                                      EMPTY))),
                           EMPTY))
```

We are given 3 functions to generate the path from the given node to the root node in a tree. Assume that each node value is unique. Only one of them is correct. Determine the outputs for the following calls, and find out which function is correct.

```
path1 EMPTY  x  _  = [ ]
path1 (NODE (a,b,c)) x lst = if ( a==x ) then (a:lst )
      else if (path1 b x (a:lst) == [ ] ) then ( path1 c x (a:lst) )
                                          else  ( path1 b x (a:lst) )

path2 EMPTY  x  _  = [ ]
path2 (NODE (a,b,c)) x lst = if ( a==x ) then (a:lst )
      else if (path2 b x (lst) == [ ] ) then ( path2 c x (lst) )
                                        else  ( path2 b x (lst) )

path3 EMPTY  x  _  = [ ]
path3 (NODE (a,b,c)) x lst = if ( a==x ) then (lst )
      else if (path3 b x (a:lst) == [ ] ) then ( path3 c x (a:lst) )
                                          else  ( path3 b x (a:lst) )
```

```
Main> path1 mytree 9 []
```

[9, 7, 5, 4, 1]

```
Main> path2 mytree 9 []
```

[9]

```
Main> path3 mytree 9 []
```

[7, 5, 4, 1]

b) (10 pts) Consider the following Haskell definitions.

```
data X  = A | B Int Y
data Y  = C | D Int X
data Z  = E X | F Y deriving Show

instance Show X where
  show (A) = "A"
  show (B a1 a2) = "B"++(show a1)++":"++(show a2)++"B"
instance Show Y where
  show (C) = "C"
  show (D a1 a2) = "D"++(show a1)++"+"++(show a2)++"D"

x_gen 0 = (A)
x_gen n = (B n (y_gen (n-1)))

y_gen 0 = (C)
y_gen n = (D n (x_gen (n-1)))

class My_Class a where
  f::a->[Int]
  f x = []

instance My_Class X where
  f (A) = []
  f (B a1 a2) = a1:(f a2)
instance My_Class Y where
  f (C) = []
  f (D a1 a2) = a1:a1:(f a2)
instance My_Class Z
```

Determine the outputs of the following Haskell function calls.

```
Main> x_gen 5
```

B5:D4+B3:D2+B1:CBDBDB

```
Main> y_gen 5
```

D5+B4:D3+B2:D1:ADBDBD

```
Main> f (x_gen 5)
```

[5,4,4,3,2,2,1]

```
Main> f (y_gen 5)
```

[5,5,4,3,3,2,1,1]

```
Main> f (F (y_gen 5))
```

[]

QUESTION 4. (15 points)

Consider the following C++ program.

- Determine its output.
- Circle the expressions in the program corresponding to dynamic binding (late binding), and **show their bindings**.

```
#include <iostream>
using namespace std;

class A{ public: int a;
             A():a(0){}
             A(int p):a(p){}
             virtual void operator+=(int p){a+=p; }
             virtual void incr(int p){a+=p; }
                     void incr2(int p){a+=2*p; }
};

class B: public A { public: int b;
             B():b(0),A(){}
             B(int p):b(p),A(2*p){}
             virtual void operator+=(int p){a+=p; b+=p; }
                     void incr(int p){a+=p; b+=p; }
                     void incr2(int p){a+=2*p; b+=2*p; }
};

class C: public B { public: int c;
             C():c(0),B(){}
             C(int p):c(p),B(2*p){}
             void operator+=(int p){a+=p; b+=p; c+=p; }
             void incr(int p){a+=p; b+=p; c+=p; }
};

void f1(A &a) { a+=10; }
void f2(A  a) { a+=10; }
void f3(B &b) { b+=10; }

main()
{
  A a1(10), *ap; B b1(20), *bp; C c1(30);
  cout<<a1.a<<"\n"<<b1.a<<" "<<b1.b<<"\n"<<c1.a<<" "<<c1.b<<" "<<c1.c<<"\n";

  ap=&b1; ap->incr(10); cout<<b1.a<<" "<<b1.b<<"\n";
  ap->incr2(10); cout<<b1.a<<" "<<b1.b<<"\n";

  f1(b1); cout<<b1.a<<" "<<b1.b<<"\n";
  f2(b1); cout<<b1.a<<" "<<b1.b<<"\n";

  bp=&c1; bp->incr(10); cout<<c1.a<<" "<<c1.b<<" "<<c1.c<<"\n";
  bp->incr2(10); cout<<c1.a<<" "<<c1.b<<" "<<c1.c<<"\n";

  f3(c1); cout<<c1.a<<" "<<c1.b<<" "<<c1.c<<"\n";
}
```

OUTPUT

```
10

40  20

120  60  30

50     30

70   30

80     40

80     40

130    70     40

150    90     40

160    100    50
```

QUESTION 5. (10 points)

Assume the following Prolog program is given:

```
pm([A], [A]).
/* [A,B|C] = [A|[B|C]]  list has at least two elements , A and B*/
pm([A, B | C ], [A|TR]) :-  pm([B|C],TR).
pm([A, B | C ], TRA) :-  pm([B|C],TR), append(TR, [A], TRA).

qA(s(A,B),s(B,A)).

qB(s(A,B),s(B,_)).

qC([X,Y|R], [Y,X|R]).

qD([1,X-1,X], [X+1|_]).

qE([X,X|R], R).
```

Give **all** answers found by Prolog for the following queries. If no solution found, write 'no':

Query	Results

pm([a,b],R)	R = [a,b]
	R = [b,a]

pm([a,b,c,d],R)	R=[a,b,c,d] R=[b,c,d,a]
	R=[a,c,d,b] R=[c,d,b,a]
	R=[a,b,d,c] R=[b,d,c,a]
	R=[a,d,c,b] R=[d,c,b,a]

qA(s(a,X),s(b,Y))	X=b, Y=a
qA(s(a,X),s(b,X))	no
qB(X,s(c,d))	X=s(_,c)
qC([1,2,3,4],R))	R=[2,1,3,4]
qD([1,2,X],R))	no
qE([Y,2,a],R))	R=[a], Y=2

QUESTION 6. (20 points)

You are asked to design a grammar for expressions of a language called Pi. Pi expressions support the following operators:

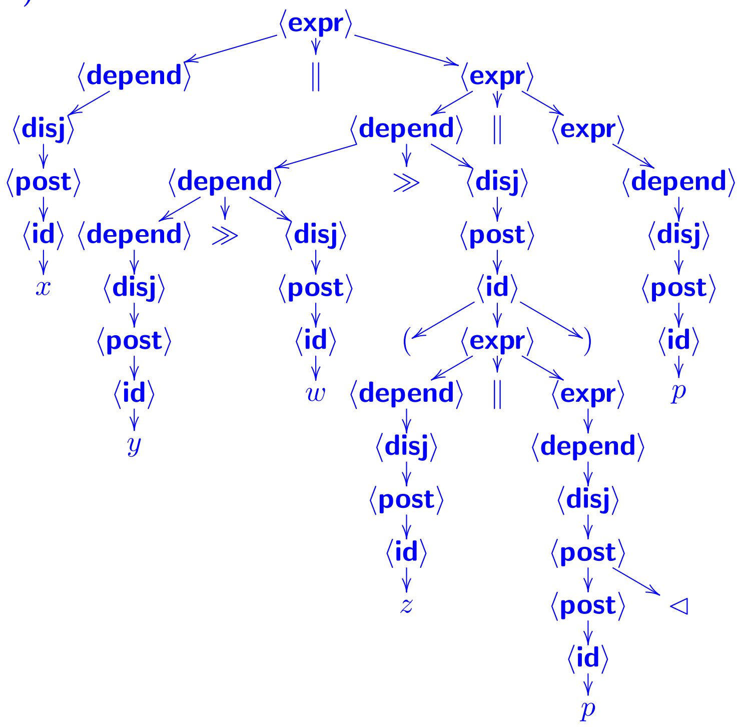
- $\|$ binary operator (concurrent evaluate)
- $\gg$ binary operator (dependent evaluate)
- $\vee$ binary operator (concurrent disjunction)
- $\triangleright$ postfix unary operator (output)
- $\triangleleft$ postfix unary operator (input)
- `(...)` paranthesis for grouping expressions

Other non-terminals of the language is letters p, t, w, x, y, z which give the basic expression. The precedence of the operators are: `(...)` has highest precedence, then $\triangleright$ and $\triangleleft$ are in the same level, then $\vee$, then $\gg$, and the lowest precedence operator is the $\|$. $\|$ is right associative, all other binary operators are left associative.
For example the expression $x \triangleright \;\|\; (y \gg z) \triangleright \vee w \vee t \;\|\; p \triangleleft$ is equivalent to:
$(x\triangleright) \;\|\; (\;(\;(\;(y \gg z)\triangleright) \vee w\;) \vee t \;\|\; (p\triangleleft)\;)$

a) Write and un-ambigous grammar that accept the sentences of this language

b) Draw the parse tree of the expression (not graded if your answer above is completely wrong):
$x \;\|\; y \gg w \gg (z \;\|\; p\triangleleft) \;\|\; p$

a) ⟨**expr**⟩ ::- ⟨**depend**⟩ | ⟨**depend**⟩ $\|$ ⟨**expr**⟩
⟨**depend**⟩ ::- ⟨**disj**⟩ | ⟨**depend**⟩ $\gg$ ⟨**disj**⟩
⟨**disj**⟩ ::- ⟨**post**⟩ | ⟨**disj**⟩ $\vee$ ⟨**post**⟩
⟨**post**⟩ ::- ⟨**id**⟩ | ⟨**post**⟩ $\triangleleft$ | ⟨**post**⟩ $\triangleright$
⟨**id**⟩ ::- p | t | w | x | y | z | (⟨**expr**⟩)
b)



METU **Department of Computer Engineering**

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Tot

CEng 242 - Programming Language Concepts

Spring 2015-2016, Final, Closed book(10 pages, 9 questions, 102 points, 150 minutes)

Name: ______________________________ **No:** __________

QUESTION 1.(12 points)

Determine the output of the following C++ program. Assume that all necessary headers and namespaces are included and all compiler optimizations are disabled.

```
class A {
    int x;
    public:
        A(int p) { x = 2*p; }
        A(const A& p) { x = 2*p.x; }
        A& operator=(const A& p) { x = 4*p.x; }
        ~A() { x = x/2; }
        int getx() const { return x; }
};

A t(2);

A f() {
    A t(2);
    return t;
}

A& h() {
    return t;
}

void g(const A &p) {
    cout << p.getx() << endl;
}

void q() {
    A a1 = A(2);
    A a2 = a1;
    a1 = a2;
    cout << a1.getx() << endl;
}

int main () {
    cout << "First output: "; g(f());
    cout << "Second output: "; g(h());
    cout << "Third output: "; q();
    return 0;
}
```

First output: 8 Second output: 4 Third output: 64

QUESTION 2.(10 points)

You are asked to implement a stock management program using C++ and object-oriented programming. The requirements are as follows:

- You must have an **abstract base class** that defines the behavior of a stock manager. Give it the name `StockManager`. This abstract base class must contain two **pure virtual** member functions, called `buy` and `sell` both of which take a **constant reference to an object** representing the historical information about the stock trades. Assume this information is of type `StockHistory`. They return an **integer** representing how many stocks to buy or sell. Complete the function prototype for the `buy` function only (assume that you are declaring this function inside the class scope):
  - int `buy(` const StockHistory& `)` = 0 ;
- This base class must also contain a **protected** member variable called `stockCount` that represents how many stocks are currently owned by us (an integer value) as well as **public accessor** and **mutator** member functions to get and set the value of this variable. Add this member variable and the related functions to this class using the correct access rights:

```
class StockManager {

protected:
        int stockCount;
public:
        int accessor() const { return stockCount;}
        void mutator(int v) { stockCount = v;}

};
```

- Assume that two new classes called `AggressiveStockManager` and `ConservativeStockManager` are derived from the `StockManager` class, both of which implement its the pure virtual functions. Answer the following questions as true (T) or false (F):
  - T Both classes can access the `stockCount` variable of their base class.
  - T We can safely assign an `AggressiveStockManager` object to a `StockManager` reference.
  - F We can safely assign an `AggressiveStockManager` object to a `ConservativeStockManager` reference.
  - F We can safely assign a `StockManager` object to an `AggressiveStockManager` reference.
  - T We cannot create instances of the `StockManager` class.

QUESTION 3.(15 points)

A new PL called METUPL is being designed and you are expected to write a **preprocessor** and **parser** for this language using Haskell. The **preprocessor** takes a SourceCode as input and produces a list of Tokens. The SourceCode and Token are defined for you as:

```
type SourceCode = String
type Token = String
```

a) Declare the type signature and implement a preprocess function which extracts tokens from the given source code and returns them a list. Note that the tokens are separated from each other only by whitespace characters but there could be multiple whitespaces between each token. For example, preprocess "   void main () " should return ["void", "main", "()"].

Make the type declaration in this box:

```
preprocess :: Sourcode -> [Token]
```

Implement the preprocess function in this box. Do not use any built-in functions (of course, you can use operators such as ++, : for list processing). If necessary implement your helper functions here or on the back of this page.

```
preprocess source = preprocess' [] source where
    preprocess' [] [] = []
    preprocess' tok [] =[tok]
    preprocess' [] (a:rest)  | a == ' ' =  preprocess' [] rest
                             | otherwise = preprocess' [a] rest
    preprocess' tok (a:rest) | a == ' ' =
           tok : preprocess' [] rest
                             | otherwise =
           preprocess' (tok ++ [a])  rest
```

b) For the **parser**, you are expected to declare a Parser typeclass. This typeclass will contain a single function called parse. This function will take two parameters with the first parameter being an **instance** of this typeclass and the second one a list of Tokens. It should return a value of ParseTree data type, whose details are given below.

- Show the definition of your type class. It must contain the type signature of the parse function as well:

```
class Parser a where
      parse ::  a -> [Token]
```

- Show the definition of the data type ParseTree. It is a possibly empty N-ary tree with Tokens represented only in the leaf nodes. You are free to choose the names of your tags.

```
data ParseTree = Node [ParseTree] | Leaf Token
```

QUESTION 4.(10 points)

Determine the output of the following C++ program.

```
class A {
public:
virtual void f() {cout<<"A::f\n";}
        void g() {f();}
virtual void h() {cout<<"A::h\n";}
virtual void i() {cout<<"A::i\n";}
};

class B:public A{
public:
void f() {cout<<"B::f\n";}
void k() {cout<<"B::k\n";}
void i() {cout<<"B::i\n";}
void j() {f();}
};

class C: public B{
public:
void f(){cout<<"C::f\n";}
void k(){cout<<"C::k\n";}
void h(){cout<<"C::h\n";}
};

void test1(A *ta) {ta->g();}
void test2(A &pa) {pa.h();}
void test3(A &pa) {pa.i();}

void test4(B *tb) {tb->j();}
void test5(B *tb) {tb->f();}
void test6(B *tb) {tb->k();}

int main(){
A a; B b; C c;

test1(&a);
test1(&b);
test1(&c);
cout<<"*****\n";
test2(c);
test3(c);
cout<<"*****\n";
test4(&c);
test5(&c);
test6(&c);
}
```

```
OUTPUT:

A::f
B::f
C::f
****
C      ::h
B      ::i
****
C::f
C::f
B      ::k
```

QUESTION 5.(10 points)

Trace the execution of the following C program and determine:

- garbage variables (GV) and dangling references (DR) (circle the statement and write as GV and DR)
- the output (fill into the table)

```
int a[2]={10,20};
int *p, *q;
int main()
{
    p=(int *)malloc(sizeof(int));
    q=a;
    *p=30;
    printf("%d %d\n",*p,*q);
    q++;
    (*q)++;
    printf("%d %d\n",*p,*q);
    p=q;        GB p
    *(a+2)=*q;  DR
    printf("%d %d\n",*p,*q);
    q=(int *)malloc(sizeof(int));
    *q=*p;
    free (q);
    (*q)++;     DR
    printf("%d %d\n",*p,*q);
}
```

OUTPUT:

30	10
30	21
21	21
21	?DR

QUESTION 6.(10 points)

Determine the output of the following program (written in a C like language) assuming static binding for the following parameter passing mechanisms:

a. lazy evaluation

b. normal order evaluation (call by name)

c. definitional mechanism (call by reference)

```
int a[4]={10,20,30,40};
int i=1;

void test(int x, int y, int z)
{
      X++;  y--;  z++;
      printf("%d %d %d\n",x,y,z);
      x++;  y--;  z++;
      printf("%d %d %d\n",x,y,z);
}
int main()
{
      test(i,a[0],a[i]);
      printf("%d %d %d %d %d\n",i,a[0],a[1],a[2],a[3]);
}
```

a. OUTPUT - lazy

```
2   9   31
3   8   32
3   8   20   32   40
```

b. OUTPUT - name

```
2   9   31
3   8   41
3   8   20   31   41
```

c. OUTPUT - reference

```
2   9   21
3   8   22
3   8   22   30   40
```

QUESTION 7.(10 points)

Determine the output of the following C++ program (some of the output is given, just determine the missing lines).

```
int i1=1, i2=2, i3=3, i4=4;

class A {
public:
        int i;
        A(int i){cout<<"A::A(int)\n"; this->i=i;}
        A(const A &a){cout<<"A::A(A)\n"; i=a.i;}
        void operator>(int &i) {cout<<"op>#1\n"; i=this->i;}
        friend void operator>(int &i, A &a) {cout<<"op>#2\n"; a.i=i;}
        friend void operator<(int &i, A &a) {cout<<"op<#1\n"; i=a.i;}
        void operator<(int &i) {cout<<"op<#2\n"; this->i=i;}
        void operator=(A &a){cout<<"A::operator=(A)\n"; a.i=i;}
};

class B:public A {
public:
        A a;
        B(int i):A(i),a(i+1){cout<<"B::B(int)\n";};
};

void f(A a1, A &a2, A *a3, A a4) {
       cout<<"f starts\n";
       a1<i1;
       i2>a2;
       (*a3)>i3;
       i4<a4;
       cout<<"f ends\n";
}

int main() {
        A a10(10), a15(15);
        B b5(5), b10=b5;

        cout<<"declarations ends\n",
        f(5, a10, &a15, b5);
        cout<<a10.i<<" "<<a15.i<<" "<<b5.i<<":"<<b5.a.i<<"\n";
        cout<<i1<<" "<<i2<<" "<<i3<<" "<<i4<<"\n";
        cout<<"assignment\n";
        b10=b5;
        cout<<b5.i<<":"<<b5.a.i<<" "<<b10.i<<":"<<b10.a.i<<"\n";
}
```

```
A::A(int)
A::A(int)
A::A(int)
A::A(int)
B::B(int)
A::A(A)
A::A(A)
declarations ends
A::A(A)
A::A(int)
f starts
op<#2
op>#2
op>#1
op<#1
f ends
2 15 5:6
1 2 15 5
assignment
A::operator=(A)
A::operator=(A)
5:6 5:6
```

QUESTION 8.(15 points)

a) Assume split /3 clause divides a list into two equal size list. Elements are distributed to first and second list on alternating order. For example split ([a,b,c,d,e,f], X, Y). gives X=[a,c,e], Y=[b,d,f]. When list has odd number of elements, first list wil get the extra element as split ([a,b,c,d,e], [a,c,e], [b,d]). Complete the split /3 as defined above:

```
split([],[],[]).          /* empty list */
split([H],[H],[]).        /* last element */
split([A|[B|R]],  [A|RA] , [B|RB] )   :-  split( R, RA, RB)
```

b) Assume merge/3 clause merges two sorted lists in ascending order into a sorted list containing elements from the both. For example merge([1,2,4,5,8],[3,7,8], R) gives R = [1,2,3,4,5,7,8,8].

```
merge([],A,A).
merge(A,[],A)  :- A=[_|_].  /* make A non-empty to eliminate ambiguity */
merge([A|ARest], [B|BRest],  [A|Result]) :- A =< B,
          merge(ARest, [B|BRest], Result),  .
merge([A|ARest], [B|BRest],  [B|Result]) :- A > B ,
          merge([A|ARest], BRest, Result),  .
```

c) Write all answers of query traverse (2,1, L). for the following Prolog program.

```
right(r).
right(e).
down(d).

traverse(3,3,[]).
traverse(X,Y,[OP|L]) :-  NX is X+1, NX =< 3, right(OP), traverse(NX,Y ,L).
traverse(X,Y,[OP|L]) :-  NY is Y+1, NY =< 3, down(OP), traverse(X ,NY,L).
```

```
L = [r,d,d]
L = [e,d,d]
L = [d,r,d]
L = [d,e,d]
L = [d,d,r]
L = [d,d,e]
```

QUESTION 9.(10 points)

Assume you are asked to define the syntax for a hypothetical page typesetting language.
Language contains the following operators:

1. The terminals of the language are capital letters. All letters from `A` to `Z` are literals describing a page id.
2. Expressions can be put in paranthesis `()` for grouping.
3. `>>`, `<<`, and `^` are unary prefix operators and describe page alignment.
4. `!` and `-` are right associative binary operators indicating current page is divided into two columns and rows respectively.
5. `\\` is a left associative binary operator indicating the page skip.
6. The precedence of operators are:
   highest is `()`, then unary alignment operators in same level, then `!` and `-` in same level, then `\\` has the lowest precedence.

a) Write an **unambigous** grammar respecting precende and associativity rules for this language. Use descriptive non-terminal names as `<aligned>` etc. Assume starting non-terminal is `<page>`.

<page> → <page> \\ <sub> | <sub>

<sub > → < align > ! <sub> | <align>

< align > → << <align> | >> <align> | ^ <align> | <simple>

<simple> → (<page>) | A | B | . . . | Z

b) Draw syntax tree of the expression ‘`>>(A\\B!^C-D\\E)!F`’.