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

NAME:_____ **ID:**_____

Question 1 : (20 pts) The following type definitions in Haskell are given:

`data X a = L1 (Int, a) | L2 (a, Int)`	$Int \times \alpha + \alpha \times Int$
`data Y a = L3 (a, (a,a))`	$\alpha \times \langle \alpha \times \alpha \rangle$
`data Z a = L4 ((a,a),a)`	$\langle \alpha \times \alpha \rangle \times \alpha$
`data W a = L5 (a,a,a)`	$\alpha \times \alpha \times \alpha$
`data M a = L6 (X a) | L7 (Y a)`	$Int \times \alpha + \alpha \times Int + \alpha \times \langle \alpha \times \alpha \rangle$
`data P a = L8 a | L9 (a, X a)`	$\alpha + \langle \alpha \times (Int \times \alpha + \alpha \times Int) \rangle$
`data Q a = L10 (a, (Int, a)) | L11 (a,(a, Int)) | L12 a`	$\alpha \times \langle Int \times \alpha \rangle + \alpha \times \langle \alpha \times Int \rangle + \alpha$
`data R a = L13 (P a, X a)`	$P\alpha \times X\alpha$
`data S a = L14 a | L15 (S a) | L16 (S a)`	$\alpha + S\alpha + S\alpha$
`data T a = L17 a | L18 (T a) | L19 (T a)`	$\alpha + T\alpha + T\alpha$
`data U a = L20 a | L21 (V a) | L22 (U a)`	$\alpha + V\alpha + U\alpha$
`data V a = L23 a | L24 (U a) | L25 (V a)`	$\alpha + U\alpha + V\alpha$

Assuming your Haskell version uses **structural equivalence** and ignores all constructor tags, which of the following types are equivalent?

X α , Y α , Z α , W α , P α , Q α , R α , S α , T α , U α , V α , X Int , M Int , X (Int,Int) , P Int , P (Int,Int) , Q Int , R Int

Note: Write as equivalence classes like:
$A \equiv B \equiv C$
D
$E \equiv F \equiv G \equiv H$
X α
Y α
Z α
W α
P α
Q α
R α
S $\alpha \equiv$ T $\alpha \equiv$ U $\alpha \equiv$ V α
X Int
M Int
X (Int,Int)
P Int
P (Int,Int)
Q Int
R Int

(P $\equiv$ Q and P Int $\equiv$ Q Int, if cartesian is assumed to be distributive over disjoint union)

Question 2 : (20 pts) **a)**Given the following C program:

```
1  int *px,*py;
2  int x;
3  int f(int *pa) {
4          int *pf;
5          int b;
6          pf=&b;
7          *px=b;
8          px=pf;
9          py=malloc(sizeof(int));
10         return *pa;
11 }
12 int *g(int a) {
13         int *pg,*ph;
14         int b;
15         pg=&b;
16         ph=malloc(sizeof(int));
17         py=&b;
18         *px=a;
19         px=pg;
20         return &x;
21 }
22 int main() {
23         int *pm;
24         pm=malloc(sizeof(int));
25         ...
26         return 0;
27 }
```

Assume your compiler wants to avoid dangling references and try to force a rule:
"Do not assign or send address of a variable to a longer lifetime pointer or variable"
at compile time. According to this rule, assume your compiler produces warnings. Give those warnings as line numbers plus a very short explanation produced by your compiler for the program above.

8: local pointer assigned to global pointer

17: local variable address assigned to global pointer

19: local pointer assigned to global pointer

b)Assume you want to add garbage collecting feature to C. Answer the following (in a couple of short sentences each):

i. Which extra information you need to keep for each variable?
(Number of references/pointers to the variable or
The list of references/pointers to the variable)

ii. What basic operations does garbage collector do in an assignment like:
px=&a;
First (decrement reference count / delete from list) of previous dereference of px, deallocate (if 0 / if empty). Then (increment reference count of a/add px to list of references to a)

iii. Does garbage collector need to do extra things for composite values (structs)? If yes what?
If structures has reference/pointer members their references need to be handled. Structure members should be recursed

Question 3 : (20 pts)

```
1  x=3
2  y=2
3  z=1
4  f a = a+1
5  let x=2
6      z=f1 x1+y1
7      f a = if a<=2 then 3
8            else (f2 (a-1))+z1
9      y=f3 x2+1
10 in  z2+f4 3
```

a)Give the line number of the binding occurrences of the following symbols, assuming the definitions between `let` and `in` are **sequentially** composed and functions are **not recursive**.

x_1: **5** x_2: **5** y_1: **2** z_1: **6** z_2: **6**

f_1: **4** f_2: **4** f_3: **7** f_4: **7**

b)Give the line number of the binding occurrences of the following symbols, assuming the definitions between `let` and `in` are **sequentially** composed and functions are **recursive**.

x_1: **5** x_2: **5** y_1: **2** z_1: **6** z_2: **6**

f_1: **4** f_2: **7** f_3: **7** f_4: **7**

c)Give the line number of the binding occurrences of the following symbols, assuming the definitions between `let` and `in` are **collaterally** composed and functions are **recursive**.

x_1: **1** x_2: **1** y_1: **2** z_1: **3** z_2: **6**

f_1: **4** f_2:**4 or 7** f_3: **4** f_4: **7**

d)Give the line number of the binding occurrences of the following symbols, assuming the definitions between `let` and `in` are **collateral-recursively** composed and functions are **recursive** (Haskell style).

x_1: **5** x_2: **5** y_1: **9** z_1: **6** z_2: **6**

f_1: **7** f_2: **7** f_3: **7** f_4: **7**

Question 4 : (20 pts) Consider the following lambda expression. Show the steps and results of its evaluation using (a) normal-order evaluation, (b) lazy evaluation, and (c) eager evaluation. +, *, / and - are the binary arithmetic operations, ^{2} is a unary operator for square, '==' is the boolean operator of equality check.

$\lambda y.\lambda x.$ **if** $(x + y == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (\lambda z.z * z \;\; 2) \quad (\lambda p.p - 4 \;\; 4)$

a) $\lambda x.$ **if** $(x + \; (\lambda z.z * z \;\; 2) \; == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (\lambda p.p - 4 \;\; 4) \mapsto$

if $(\; (\lambda p.p - 4 \;\; 4) \; + \; (\lambda z.z * z \;\; 2) \; == \; (\lambda p.p - 4 \;\; 4)^2)$

then $\; (\lambda p.p - 4 \;\; 4)^2 + \; (\lambda p.p - 4 \;\; 4)$ **else** $2/\; (\lambda p.p - 4 \;\; 4)$ **endif** $\quad \mapsto$

if $(\; (4 - 4) \; + \; (\lambda z.z * z \;\; 2) \; == \; (\lambda p.p - 4 \;\; 4)^2)$ **then** ... **else** ... **endif** $\quad \mapsto$

if $(\; (4 - 4) \; + \; 2 * 2) \; == \; (\lambda p.p - 4 \;\; 4)^2)$ **then** ... **else** ... **endif** $\quad \mapsto$

if $(\; (4 - 4) \; + \; 2 * 2) \; == \; (4 - 4)^2)$ **then** ... **else** ... **endif** $\quad \mapsto$

if $false$ **then** ... **else** ... **endif** $\quad \mapsto 2/(\lambda p.p - 4 \;\; 4) \; \mapsto$

$2/(4 - 4) \; \mapsto 2/0 \mapsto error!$

b) $\lambda x.$ **if** $(x + \;$ y:$(\lambda z.z * z \;\; 2) \; == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (\lambda p.p - 4 \;\; 4) \mapsto$

if $(\;$ x:$(\lambda p.p - 4 \;\; 4) \; +$ y:$(\lambda z.z * z \;\; 2) \; ==$ x:$(\lambda p.p - 4 \;\; 4)^2)$
then x:$(\lambda p.p - 4 \;\; 4)^2 +$ x:$(\lambda p.p - 4 \;\; 4)$ **else** $2/$x:$(\lambda p.p - 4 \;\; 4)$ **endif** $\quad \mapsto$

if $(\;$ x:$(4 - 4) \; +$ y:$(\lambda z.z * z \;\; 2) \; ==$ x:$(\lambda p.p - 4 \;\; 4)^2)$
then x:$(\lambda p.p - 4 \;\; 4)^2 +$ x:$(\lambda p.p - 4 \;\; 4)$ **else** $2/$x:$(\lambda p.p - 4 \;\; 4)$ **endif** $\quad \mapsto$

if $(\;$ x:$0 \; +$ y:$(\lambda z.z * z \;\; 2) \; ==$ x:$0^2)$ **then** x:$0^2 +$ x:0 **else** $2/$x:0 **endif** $\quad \mapsto$

if $(\;$ x:$0 \; +$ y:$(2 * 2) \; ==$ x:$0^2)$ **then** x:$0^2 +$ x:0 **else** $2/$x:0 **endif** $\quad \mapsto$

if $false$ **then** x:$0^2 +$ x:0 **else** $2/$x:0 **endif** $\quad \mapsto$

$2/$x:$0 \; \mapsto error!$

c) $\lambda y.\lambda x.$ **if** $(x + y == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (2 * 2) \quad (\lambda p.p - 4 \;\; 4) \mapsto$

$\lambda x.$ **if** $(x + 4 == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (\lambda p.p - 4 \;\; 4) \mapsto$

$\lambda x.$ **if** $(x + 4 == x^2)$ **then** $x^2 + x$ **else** $2/x$ **endif** $\quad (4 - 4) \mapsto$

if $(0 + 4 == 0^2)$ **then** $0^2 + 0$ **else** $2/0$ **endif** $\; \mapsto$

if $false$ **then** $0^2 + 0$ **else** $2/0$ **endif** $\; \mapsto$

$2/0 \mapsto error!$

Question 5 : (20 pts) Suppose that we want to add some parametric polymorphism to C (not to C++). The goal of the designers is to have more abstraction in C programs, e.g. `t binop (t x, t y){...}` for a function `binop` to do some binary operation for any type `t`. Similarly, something like `t id (a x, b y, c z, d q){...}` may return one of the formal parameters, where `t` is supposed to range over possible formal parameter types, which can be any polytype `a,b,c,d`.

a) What constructions (definitions, declarations etc.) would you add to C to incorporate this change? You don't need to be exhaustive or complete. Just give us an idea how you would proceed.

b) Can we do something similar without introducing parametric polymorphism to C, using the currently available declarative and definitional properties of the language? Briefly comment to what extent this can be done.