METU, Department of Computer Engineering
CENG 242 - PROGRAMMING LANGUAGES CONCEPTS
MID TERM EXAM (Spring 2006)
CLOSED NOTES AND BOOKS, 105 pts, DURATION: 120 mins

NAME:______________________________________ **ID:**_____________

QUESTION 1. (15 pts)

For the given following datatype definitions write two Haskell functions called *f* and *g* to convert *Tavuk* and *Yumurta* typed values into integers as folows: for each *A*, *B*, *C*, *D*, *E*, and *F* in an *Tavuk* or *Yumurta* value, add 1, 2, 3, 4, 5, and 6 respectively. For example, if a *Tavuk* value contains 2 *C*'s, one *A*, one *E*, and one *D*, then *f* should produce 16 (i.e. f (C (C (A (D E)))) ⇒ 3+3+1+4+5 ⇒ 16).

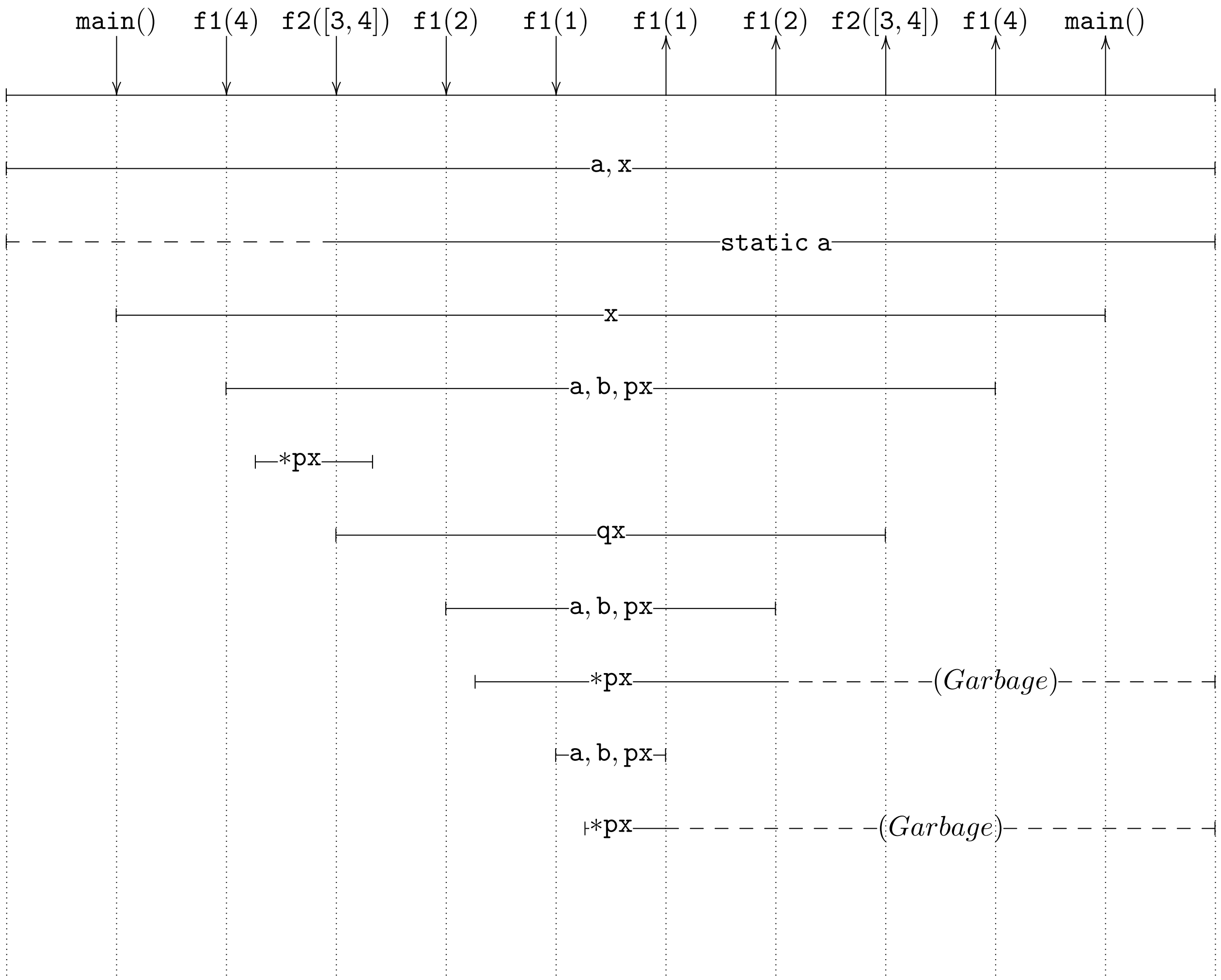
```
data Tavuk = A Yumurta | B | C Tavuk
  deriving Show
data Yumurta = D Yumurta | E | F Tavuk
  deriving Show

Solution:
----------

f (A x) = 1 + g x
f B     = 2
f (C x) = 3 + f x

g (D x) = 4 + g x
g E     = 5
g (F x) = 6 + f x
```

Do **not** define any auxiliary functions in the implementations.

QUESTION 2. (20 pts)

Show the lifetimes of the variables in the following C program. In order to do this, you need to trace the execution of the program until its termination. Use a time-chart to show when the variables are created and destroyed related to the functions' executions (i.e., call and return).

```
#include <stdio.h>
int a=2;
int x=1;

int f2(int*);

int f1(int a)
{
    int b=x;
    int *px;
    printf("ENTER f1 %d\n",a);
    px=(int *) malloc(2*sizeof(int));
    *px = a-b;
    px[1]=a;
    if ((*px>0) && (*(px+1)<3))
       *px=f1(px[0]);
    if (*px>1)
       b=f2(px);
    return (b);
}

int f2 (int *qx)
{
    static int a=2;
    printf("ENTER f2 %d\n",*qx);
    if (qx)
      free (qx+1); free(qx);
    a=f1(a);
    return (a);
}

main()
{
    int x=2;
    f1(a+x);
}


// The output of the program is as follows:
// ENTER f1 4
// ENTER f2 3
// ENTER f1 2
// ENTER f1 1
```


main()
f1(4)
f2([3,4])
f1(2)
f1(1)
f1(1)
f1(2)
f2([3,4])
f1(4)
main()
a, x
static a
x
a, b, px
*px
qx
a, b, px
*px
(Garbage)
a, b, px
*px
(Garbage)

QUESTION 3. (15 pts)

Determine the environments of the required positions of the following C program.

```
#include <stdio.h>
int a=2;
int x=1;

int f2(int*);

int f1(int a)
{
    int b=x;
    int *px;       // Environment = { a:param, x:global, f2:func, f1:func,
    printf("ENTER f1 %d\n",a);            //                  b:int, px:int*}
    px=(int *) malloc(2*sizeof(int));
    *px = a-b;
    px[1]=a;
    if ((*px>0) && (*(px+1)<3))
       *px=f1(px[0]);
    if (*px>1)
       b=f2(px);
    return (b);
}

int f2 (int *qx)
{
    static int a=2;      // Environment = {a:static, x:global, f2:func,
    printf("ENTER f2 %d\n",*qx);        //            f1:func, qx:int*}
    if (qx)
      free (qx+1); free(qx);
    a=f1(a);
    return (a);
}

main()
{
    int x=2;    // Environment = {a:global, x:int, f2:func, f1:func, main:func}
    f1(a+x);
}
```

QUESTION 4. (20 pts)

Assume you have the following code written in an extended C version that allows functions to be declared inside of a function. These functions will have a local scope as the local variables have.

```
int min=5, max=10;

int check(int i) {
     if (i<max)  return max-i;
     else        return i-min;
}

int testit(int n) {
     int min=2;

     if (n>=min && n<=max)  return check(n);
     else                   return check(2*n);
}

int main() {
    int max=20;
    int check(int i) {
         return min+max-i;
    }

    printf("%d\n",testit(15));
    printf("%d\n",check(12));

    return 0;
}
```

Note that the function `check()` is called twice at run time. First is via `testit()` and the second is directly from `main()`.

a) Assume the language uses static scoping/binding. For these two calls of `check()`, which binding occurences bind `min` and `max` in the function body? (answer as `global, check, testit, main`)

```
first call:  min:  global
first call:  max:  global
second call:  min:  global
second call:  max:  main
```

b) Assume the language uses static scoping/binding. what is the output of the program?

```
25
13
```

c) Assume the language uses dynamic scoping/binding. For these two calls of `check()`, which binding occurences bind `min` and `max` in the function body? (answer as `global, check, testit, main`)

```
first call:  min:  tesit
first call:  max:  main
second call:  min:  global
second call:  max:  main
```

d) Assume the language uses dynamic scoping/binding. what is the output of the program?

```
7
13
```

QUESTION 5. (15pts)

You are given the following overloaded versions of a function in a C like language:

double x;
int i;
double f^1(double n) { .... }
int f^2(int n) { .... }
double f^3(int n) { .... }

Assume your language only allows int to double coercion (implicit type conversion), not the other way. '+' operator is overloaded as int$\times$int$\rightarrow$int and double$\times$double$\rightarrow$double only. Let an unambiguous interpretation in this language mean all overloadings are resolved and all coersions applied explicitly, like: f^1((double) 5)+f^2(4)

a) Assuming language applies context insensitive overloading and only f^1 and f^2 exist. What are the all possible unambiguous interpretations of the following two expressions:

x=f(i)+f(x);

x=f^1((double)i)+f^1(x)   x=(double)f^2(i)+f^1(x)

x=f(f(i))+f(x);

x=f^1(f^1((double)i))+f^1(x)   x=f^1((double)f^2(i))+f^1(x)   x=(double)f^2(f^2(i))+f^1(x)

b) Assuming language applies context sensitive overloading and all functions above exist. What are the all possible unambiguous interpretations of the following two expressions:

x=f(i)+f(x);

x=f^1((double)i)+f^1(x)   x=(double)f^2(i)+f^1(x)   x=f^3(i)+f^1(x)

x=f(f(i))+f(x);

x=f^1(f^1((double)i))+f^1(x)   x=f^1((double)f^2(i))+f^1(x)   x=(double)f^2(f^2(i))+f^1(x)
x=f^3(f^2(i))+f^1(x)   x=f^1(f^3(i))+f^1(x)

c) What is the most general type (inferred type by the Haskell) of the following Haskell function misery:

```
data Tree a = Leaf a | Branch (a,Tree a,Tree a)

misery (Leaf x) f s = (1, f x s)
misery (Branch(x,t1,t2)) f s =
        let (c,r)=misery t2 f s
            (d,q)=misery t1 f r
        in
            (c+d+1,f x q)
```

Tree $\alpha \rightarrow (\alpha \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow$ (Int $\times\ \beta$)

QUESTION 6. (20 points)

Determine the output of the following program (written in a C like language) for the following parameter passing mechanisms:
a) definitional mechanism, variable parameter (call by reference)
b) copy mechanism, value parameter
c) copy mechanism, value-result parameter
d) call by name (normal order evaluation)

```
int x=12,y=10;

void tswap(int pa, int pb) {
  int tmp;
  tmp=pa;
  pa=pb;
  pb=tmp;
  x=x+pa;
  x=x-pb;
  y++;
  printf("%d %d %d %d\n",pa,pb,x,y);
}

int main() {
  int a=4;
  tswap(x,a);
  printf("%d %d %d\n",x,y,a);

  tswap(++x,++y);
  printf("%d %d %d\n",x,y,a);
  return 0;
}
```

Assume ++x increments the variable and then gives the reference of the variable. In other words, it can be used as an l-value.

a)

```
-4 12 -4 11
-4 11 12
27 -2 27 -2
27 -2 12
```

b)

```
4 12 4 11
4 11 4
12 5 12 13
12 13 4
```

c)

```
4 12 4 11
4 11 12
12 5 12 13
12 5 12
```

d)

```
-4 12 -4 11
-4 11 12
29/30 0 28/29/30 -1/0/0
30/29 0 12
```

(due to ambiguity from printf)