**CS5363 Programming Languages and Compilers: Sample Questions**

**Time: 7:30pm to 8:45pm, May 2nd**

**Name (Please Print): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PART I Multiple Choice (8\*6 points = 48 points)**

1. Consider the following grammar and type inference rules. What is the type of:

(0, 0), (0, 0)

Grammar:

S -> true | 0 | (T)

T -> T, S | S

Type inference rules (type1 indicates any type):

|-0 : int |-true: bool

|-T: type1 |-T: type1 && |-S: type1 |-T: type1 && |-S: bool

-------------------- ------------------------------- -----------------------------

|-(T): (type1) |-T, S: type1 |-T, S: type1

A. int

B. (int)

C. (int, int)

D. ((int))

Answer: B

2. Which of the following expression has a type error according to the type inference rules in Question 1?

A. (0, true), (0, true)

B. (true, 0), (true, 0)

C. (0, 0, 0)

D. 0, true, true

Answer: B

3. Which common expression(s) are available at the program point X?

y = log(a+b)

while (y < a\*b){

-------------------------------------------------X

 b = a + b;

 y = b \* b;

}

A. a\*b

B. a\*b, a+b

C. a\*b, b\*b

D. None of the above

Answer: A

4. Which of the following code has the potential to be optimized with loop unrolling?

A. for (int i = 0; i <10; i++) { if(a[i] < 0) return; }

B. for (int i = 0; i <10; i++) { a[i] = a[i+1] + 1;}

C. for (int i = 0; i <10; i++) { i = i + a[i];}

D. while (i<0) { i++; }

Answer: B

**PART II Short Answer Questions (52 points)**

1. Given 3-address code below, please use liveness analysis and graph coloring to find out whether it is possible to generate machine code with 4 registers without spilling data out to memory (13 points).

add r1, r2 => r3 {r1, r2, r4}

mult r2, r4 => r5 {r2, r3, r4}

sgt r5, r3 => r6 {r3, r5}

JZ r6 => L1 {r5, r6}

add r5, 1 => r7 {r5}

sub r7, r5 => r8 {r5, r7}

L1:

writeINT r5 {r5}

4 colors: R, G, B, Y

r1 -> R

r2 -> G

r3 -> R

r4 -> B

r5-> G

r6 -> B

r7 -> B

Correct as long as simultaneously live variables are not assigned same color.

2. Please give the type of functions f4, f5 and f6, and explain what they do (13 points).

f4 [] x = []

f4 (y:ys) x = if x == y then f4 ys x else (y:f4 ys x)

f5 ls x = x:f4 ls x

f6 ls = foldl f5 [] ls

Answer:

f4: [a] -> a -> [a]

f5: [a] -> a -> [a]

f6: [a] -> [a]

f4: remove x from ls

f5: remove x from ls and append x at the beginning of ls

f6: remove all duplicated elements in ls