Instructions

1. The is a closed-book exam, but you are allowed to use a single page (both sides) of notes.
2. There are 6 problems in this exam and you have 75 minutes to answer them.
3. To receive full credit, your solution must not only be correct but also show all the steps.
4. Discussing the exam contents with anyone who has not taken the exam is a violation of the academic honesty code.

Problem 1. (10 points) Consider the following *j*-- program:

```
public class Mystery {
    private int f = 42;

    public static int f(int u, long v, int w) {
        int x = u + w;
        {
            Mystery m = new Mystery();
            long y = m.g(u, v);
            long z = (long) x * y;
        }
        {
            Mystery m = new Mystery();
            int y = u * w;
            long z = m.g(w, v);
        }
        return x;
    }

    public long g(int u, long v) {
        int w = u * u;
        long x = (long) f + (long) w;
        long y = x * x;
        return y;
    }
}
```

a. (5 points) Show in detail the compilation unit context for `Mystery`, as built by `preAnalyze()`.

b. (5 points) Show in detail the method/local contexts for `f()` and `g()`.

Problem 2. (10 points) Classify the following casts as identity, narrowing, widening, boxing, or unboxing. What instructions (if any) must generated for each?

a. (1.25 points)

```
String x = (String) (new Object());
```

b. (1.25 points)

```
char x = (char) 42;
```

c. (1.25 points)

```
char x = (char) (new Character('*'));
```

d. (1.25 points)

```
Object x = (Object) (new String("Hello, World"));
```

e. (1.25 points)

```
Boolean x = (Boolean) false;
```

f. (1.25 points)

```
String x = (String) (new String("Hello, World"));
```

g. (1.25 points)

```
int x = (int) '@';
```

h. (1.25 points)

```
char x = (char) '@';
```

Problem 3. (20 points) Suppose `o` is an object and `f` an instance integer field within, `a` is an array of integers, and `y`, `i` and `z` are integers. For each of the *j*-- statements below, using the following table, list the instructions generated and show how the runtime stack evolves as those instructions are executed.

	x	a[i]	o.f	C.sf
lhs = y	iload y' [dup] istore x'	aload a' iload i' iload y' [dup_x2] iastore	aload o' iload y [dup_x1] putfield f	iload y' [dup] putstatic sf
lhs += y	iload x' iload y' iadd [dup] istore x'	aload a' iload i' dup2 iaload iload y' iadd [dup_x2] iastore	aload o' dup getfield f iload y' iadd [dup_x1] putfield f	getstatic sf iload y' iadd [dup] putstatic sf
++lhs	iinc x',1 [iload x']	aload a' iload i' dup2 iaload iconst_1 iadd [dup_x2] iastore	aload o' dup getfield f iconst_1 iadd [dup_x1] putfield f	getstatic sf iconst_1 iadd [dup] putstatic sf
lhs--	[iload x'] iinc x',-1	aload a' iload i' dup2 iaload [dup_x2] iconst_1 isub iastore	aload o' dup getfield f [dup_x1] iconst_1 isub putfield f	getstatic sf [dup] iconst_1 isub putstatic sf

a. (10 points)

```
o.f--;
```

b. (10 points)

```
z = ++a[i];
```

Problem 4. (20 points) Consider the following JVM bytecode for a *j--* method `int mystery(int x, int y)`:

```
public static int mystery(int, int);
        0: iconst_1
        1: istore_2
        2: iload_1
        3: iconst_0
        4: if_icmple     18
        7: iload_2
        8: iload_0
        9: imul
       10: istore_2
       11: iload_1
       12: iconst_1
       13: isub
       14: istore_1
       15: goto          2
       18: iload_2
       19: ireturn
```

a. (10 points) Show how the runtime stack evolves as a result of the call `mystery(3, 4)`.

b. (5 points) What value does the above call return?

c. (5 points) What does `mystery(x, y)` compute and return in general?

Problem 5. (20 points) The LIR instructions for the method `int mystery(int x, int y)` from the previous problem are listed below.

```
B0

B1
0: LDC [1] [V32|I]
5: MOVE $a1 [V33|I]
10: MOVE [V32|I] [V34|I]

B2
15: LDC [0] [V35|I]
20: BRANCH [LE] [V33|I] [V35|I] B4

B3
25: MUL [V34|I] $a0 [V36|I]
30: LDC [1] [V37|I]
35: SUB [V33|I] [V37|I] [V38|I]
40: MOVE [V36|I] [V34|I]
45: MOVE [V38|I] [V33|I]
50: BRANCH B2

B4
55: MOVE [V34|I] $v0
60: RETURN $v0
```

a. (10 points) Compute the liveUse and liveDef sets for each basic block in the method.

b. (10 points) Compute the liveIn and liveOut sets for each basic block in the method.

Problem 6. (20 points) The liveness intervals for the registers (all except v33 and v34) in the LIR instructions from the previous problem are listed below.

```
 v0: [55, 60]
 a0: [0, 25] [25, 50]
 a1: [0, 5]
V32: [0, 10]
V33: ???
V34: ???
V35: [15, 20]
V36: [25, 40]
V37: [30, 35]
V38: [35, 45]
```

a. (10 points) Compute the liveness intervals for v33 and v34.

b. (5 points) List the neighbors of the 7 vertices (v32 — v38) in the interference graph G for the method, in ascending order.

c. (5 points) Color the graph G using 3 physical registers and draw the graph using symbols ▲, ★, and ■ for vertices, where the symbols denote the three registers.

Exam 2 Solutions

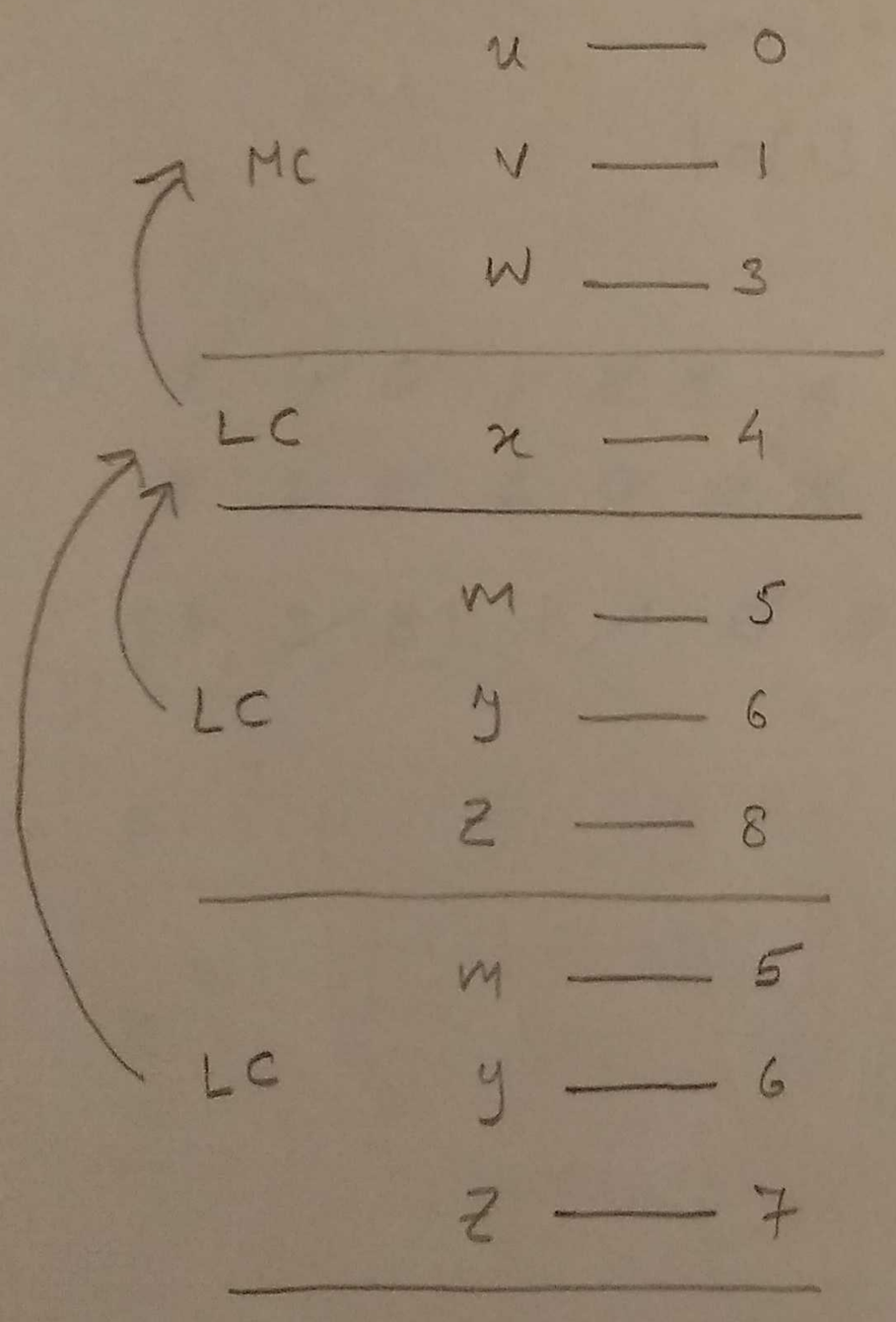
(1) a) java.lang.String
String
→ classfile → String.class

java.lang.Object
Object
→ classfile → Object.class

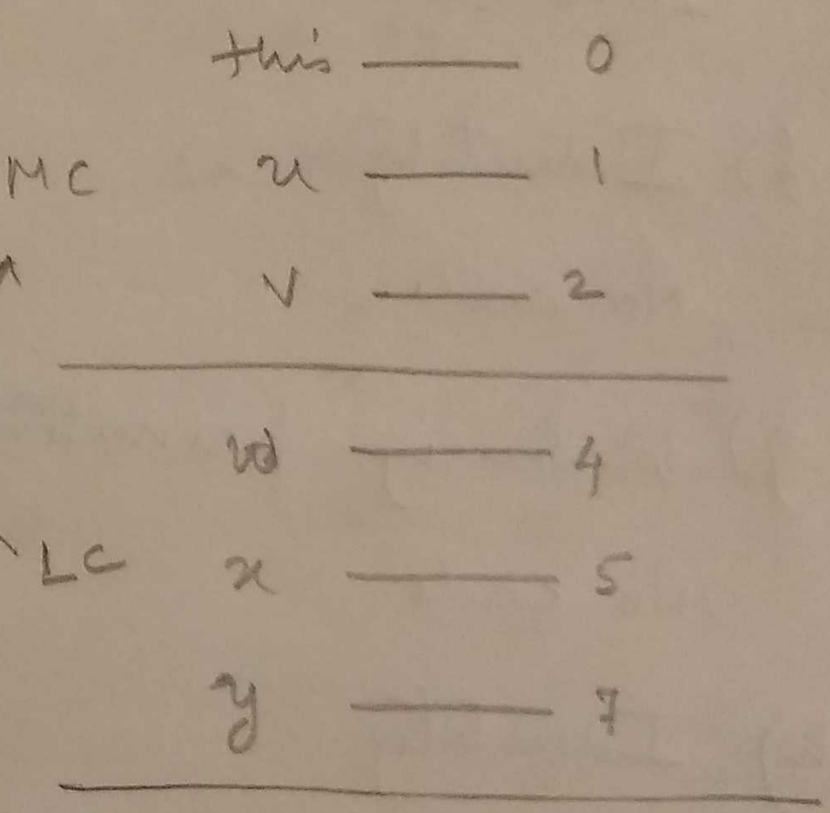
Mystery → classfile → Mystery.class

- private int f;
- public static int f(int, long, int)
- public long g(int, long)

b) f()



g()



MC ⇒ Method Context

LC ⇒ Local Context

(2) a) Narrowing reference

CHECKCAST java/lang/String (fails at runtime)

b) Narrowing primitive

I2C

c) Unboxing

INVOKEVIRTUAL Character.charValue (Ljava/lang/Character;)C

d) Widening reference

No code

e) Boxing

INVOKESTATIC Boolean.valueOf (Z)Ljava/lang/Boolean;

f) Identity

No code

g) Widening primitive

No code

h) Identity

No code

(3) a) o.f--

|o	aload 0
|o|o	dup
|o|o.f	getfield f
|o|o.f|1	iconst_1
|o|o.f-1	isub
|	putfield f

b) z = ++a[i];

|a	aload a
|a|i	iload i
|a|i|a|i	dup2
|a|i|a[i]	iaload
|a|i|a[i]|1	iconst_1
|a|i|a[i]+1	iadd
|a[i]+1|a|i|a[i]+1	dup_x2
|a[i]+1	iastore

(4) a)

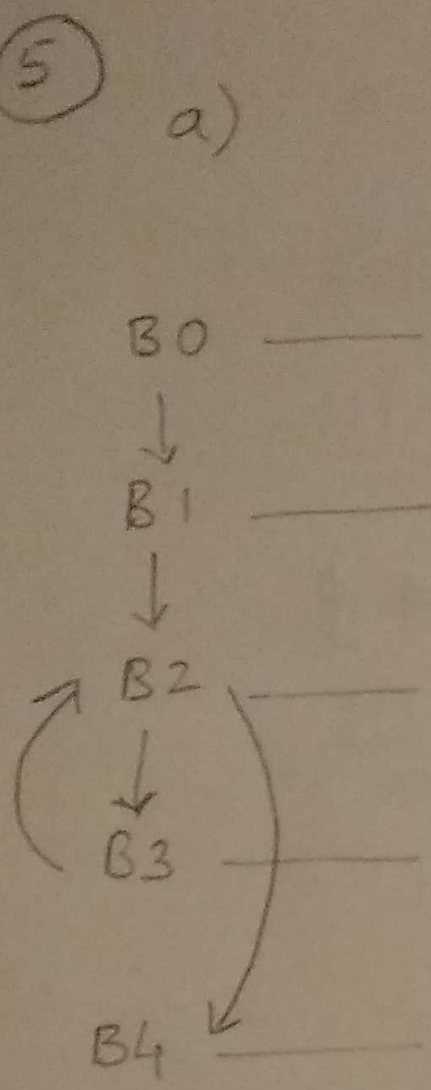
0	1	2
x	y	z
3	4	
	3	1
	2	3
	1	9
	0	27
		81

b) 81

c) x^y

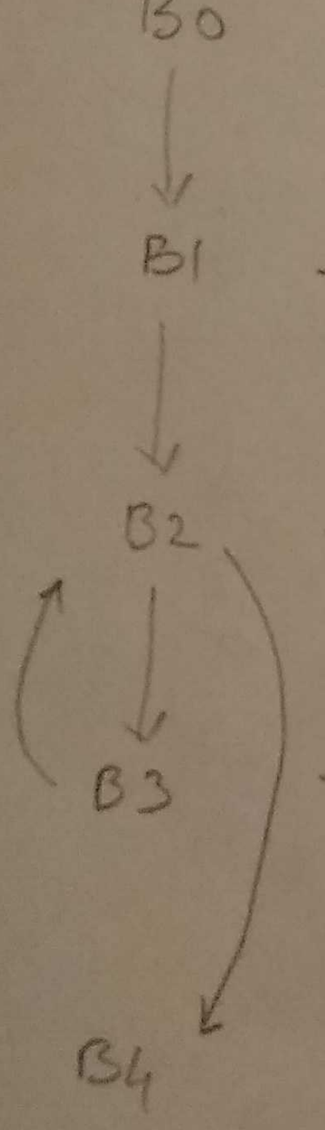
(5)

a)



	liveUse	liveOut
B0	—	—
B1	$a1	V32, V33, V34
B2	V33	V35
B3	$a0, V33, V34	V33, V34, V36, V37, V38
B4	V34	$V0

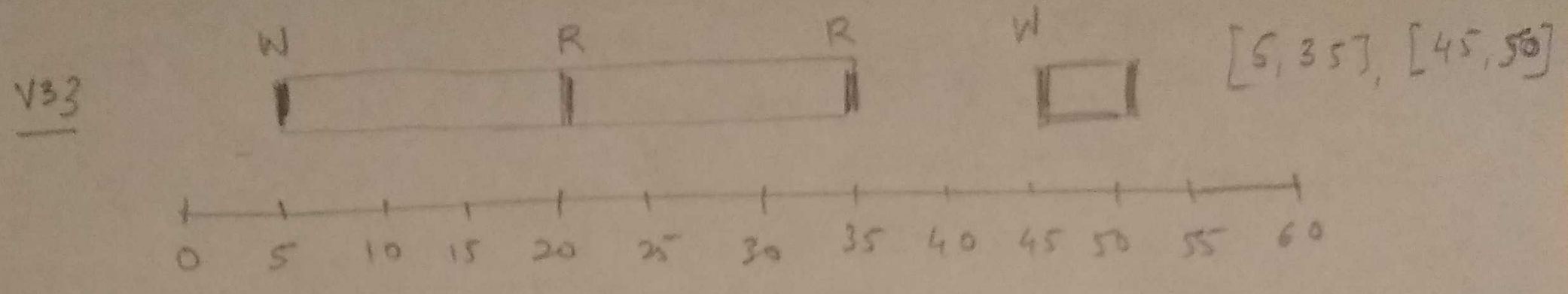
b)



	liveIn	liveOut
B0	$a0, $a1	$a0, $a1
B1	$a0, $a1	$a0, V33, V34
B2	$a0, V33, V34	$a0, V33, V34
B3	$a0, V33, V34	$a0, V33, V34
B4	V34	—

(6) a)

(3)



b)

V32 —— V33

V33 —— V32, V34, V35, V36, V37

V34 —— V33, V35, V38

V35 —— V33, V34

V36 —— V33, V37, V38

V37 —— V33, V36

V38 —— V34, V36

c) Allocation

V32 — r1

V33 — r2

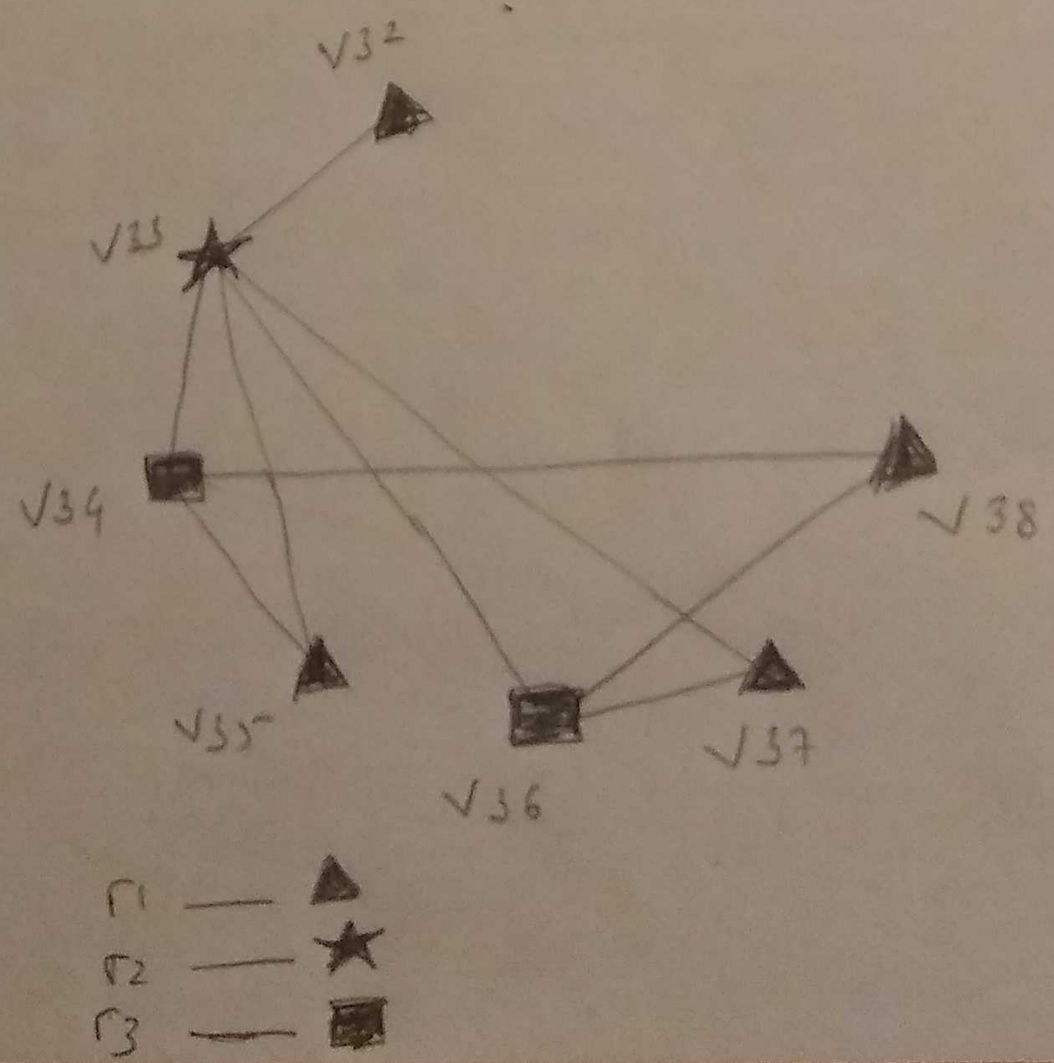
V34 — r3

V35 — r1

V36 — r3

V37 — r1

V38 — r1



Stack

~~V32~~

~~V35~~

~~V37~~

~~V38~~

~~V33~~

~~V34~~

~~V36~~