Compiler Construction
Professor Jeremy Siek

Quiz 4
October 15, 2012

Name: ____________________

This exam has 3 questions, for a total of 10 points.

1. $\boxed{\text{3 points}}$ In several places within the Explicate pass, your compiler generates code that checks whether the result on an expression is true-like according to Python's semantics. Write down a helper function that takes as input a P_1 AST node for a pure expression (no side effects) and returns an "explicit AST" that implements the true-like semantics of Python, producing an integer at runtime with 0 representing false and 1 representing true. You may use the new AST classes described in Chapter 4 that are part of the "explicit AST", which includes `Const`, `Name`, `Add`, `UnarySub`, `IfExp`, `InjectFrom`, `ProjectTo`, `GetTag`, `Compare`, `Let`, and `CallFunc`.

Solution:

```
def gen_is_true(e):
    return IfExp(Compare(GetTag(e), [('==', Const(tag['big']))]),
                 CallFunc(Name('is_true'), [e]),
                 Compare(Const(0), [('!=', ProjectTo('int', e))]))
```

2. $\boxed{\text{3 points}}$ What is the output of the following P_1 program?

```
x = [[1],[2]]
y = x[1]
z = y
y[0] = 0
v = x + []
x = x + x
w = x + v
print w
```

Solution:

```
[[1], [0], [1], [0], [1], [0]]
```

Name: ____________________

3. $\boxed{\text{4 points}}$ Write down the x86 assembly code that your compiler would produce for the following P_1 program.

```
print (True if input() else 42)
```

Solution: The following is one of many correct solutions.

```
.globl main
main:
   pushl %ebp
   movl %esp, %ebp
   subl $8, %esp
   call input_int       # input()
   movl %eax, %ebx
   movl %ebx, %eax
   andl $3, %eax        # big pyobj?
   cmpl $3, %eax
   sete %al
   movzbl %al, %ecx
   cmpl $0, %ecx
   je label_10_else
   pushl %ebx
   call is_true        # yes a big pyobj, call is_true
   addl $4, %esp
   jmp label_11_if_end
label_10_else:         # not a big pyobj
   sarl $2, %ebx
   cmpl $0, %ebx
   setne %al
   movzbl %al, %ebx
   movl %ebx, %eax
label_11_if_end:
   cmpl $0, %eax      # check result of is-true
   je label_12_else
   movl $1, %eax      # then branch, produce True
   sall $2, %eax      # inject to pyobj
   orl $1, %eax
   jmp label_13_if_end
label_12_else:
   movl $42, %eax     # else branch, produce 42
   sall $2, %eax      # inject to pyobj
   orl $0, %eax
label_13_if_end:
   pushl %eax
   call print_any     # print
   addl $4, %esp
   movl $0, %eax
   leave
   ret
```