This exam has 4 questions, for a total of 10 points.

1. **2 points**  What is the output of the following $P_3$ program?

   ```python
   class A:
       def f(self):
           return self.g()
       def g(self):
           return 2
   class B(A):
       def g(self):
           return 1
   a = A()
b = B()
print a.f()
print b.f()
   ```

**Solution:** One point per line of output:

   2
   1

2. **2 points**  What is the output of the following $P_3$ program?

   ```python
   class C:
       def __init__(self, n):
           self.n = n
   C.x = lambda a: a.n * a.n
   def m(f,l,i):
       return [] if i == 0 else m(f,l,i - 1) + [f(l[i-1])]
l0 = [1,1,2,3,5]
l1 = m(lambda x: x + 1, l0, 5)
print l1
l2 = m(lambda x: C(x).x, l1, 5)
l3 = m(lambda f: f(), l2, 5)
print l3
   ```

**Solution:** One point per line of output:

   [2, 2, 3, 4, 6]
   [4, 4, 9, 16, 36]
3. 2 points  What is the output of the following $P_3$ program?

```python
x = 1
class C:
    x = 2
    def f(self):
        print x
    if True:
        x = 3
    print x
C.f(C())
```

**Solution:** One point per line of output:

```
3
1
```

4. 4 points  Fill in the cases for `Name` and `Class` AST nodes in the following `declassify` function. The `cls` parameter should be the temporary name for a class, when inside a class body, otherwise it should be `None`. Your answer should demonstrate that you were awake during the lecture.

```python
def declassify(n, cls):
    if isinstance(n, Module):
        return Module(n.doc, declassify(n.node, None))
    elif isinstance(n, Stmt):
        return Stmt([declassify(s, cls) for s in n.nodes])
    elif isinstance(n, UnarySub):
        return UnarySub(declassify(n.expr, cls))
    elif isinstance(n, Name):
    elif isinstance(n, Class):
        tmp = generate_name(n.name)
        bases = [declassify(b, cls) for b in n.bases]
        code = declassify(n.code, tmp)
```
**Solution:** 2 points for the `Name` case and 2 points for the `Class` case.

```python
e elif isinstance(n, Name):
    if cls:
        return IfExp(CallFunc(Name('has_attr'),
                              [Name(cls), Const(n.name)]),
                   CallFunc(Name('get_attr'),
                             [Name(cls), Const(n.name)]),
                   n)
    else:
        return n

e elif isinstance(n, Class):
    tmp = generate_name(n.name)
    bases = [declassify(b, cls) for b in n.bases]
    code = declassify(n.code, tmp)
    tmp_assign = Assign([AssName(n.name, 'OP_ASSIGN')], Name(tmp))
    return Stmt([Assign([AssName(tmp, 'OP_ASSIGN')],
                     CallFunc(Name('create_class'), [List(bases)])) + 
                  [code] + 
                  [tmp_assign])
```