Department of Computer Science

CSC 326H1F — Fall 2006

Test 2

Aids allowed: one 2-sided sheet

Time: 50 minutes

Total marks: 30

name: _____

student number: _____

1	
2	
3	
Total	

1. [10 marks = 5 + 5]

(a) [5 marks]

Write a Python program that takes as input a Python program and uses regular expressions to find and print the names of functions defined in the input program. For example, if the input is the program in question 2, the output would be:

```
mul
up1
up2
down1
down2
stop
```

Assume that function names contain only alphanumeric characters. They differ from actual Python names in two ways:

- Underscores are not accepted.
- The first character may be a digit.

Your program must not be thrown off track by white space anywhere it might appear, even if a function is within an indented block of code such as a class.

You can assume the input program is in the standard input.

1. (continued)

(b) [5 marks]

Write a program like the one in part (a), but intended to find all the function *calls* in the input. For example, if the input is the program in question 2, the output would be:

```
mul(x, 4)
mul(x, 3)
mul(x, 4)
mul(x, 3)
mul(x, 3)
f(x)
```

Clarifications:

- Function names are the same as in part (a).
- You may assume any name before parentheses, such as the `f` in `f(…)`, is a function name.
- There may be multiple function calls in an input line. In the output, each call must be on a separate line.
- Function *definitions* should be omitted.
- You may assume that there are no parentheses inside strings or comments.
- There must be no white space in the output except within parentheses.
- Within parentheses, leave white space unchanged.

2. [15 marks = 5 + 10]

Below is a Python program that runs without error.

```
def mul(x, f):
    # a % b is the remainder when a is divided by b.
    return x % f == 0

def up1(x):
    if not mul(x, 4):
        return x + 3, up1
    elif mul(x, 3):
        return x + 2, up2
    else:
        return x, down1

def up2(x):
    if mul(x, 4):
        return x, down2
    else:
        return x + 2, up2

def down1(x):
    if mul(x, 3):
        return x, up2
    else:
        return x - 1, down1

def down2(x):
    if x == 0:
        return x, stop
    elif mul(x, 3):
        return x, up2
    else:
        return x - 4, down2

def stop(x):
    return x, stop

x = 2
f = up1
while f != stop:
    print x,
    x, f = f(x)
print ''
```

(a) [5 marks]

What is the output from the program?

Hint: There is one line of output, containing fewer than twenty numbers.

2. (continued)

(b) [10 marks]

Write a Scheme program that does the same things as the Python program. In place of the Python code that is not in a function, write a Scheme function `q1` that can be called with no arguments to run the program.

Requirements and hints:

- Don't be afraid to try this part even if you couldn't do part (a).
- You must have one function in Scheme for every function in Python, and the names must be the same.
- You may write helper functions.
- To return a pair of items in Scheme, you will need to make a list of two items. Use "`list`" or "`cons`" to build lists.
- In Scheme, `(remainder a b)` returns the remainder when `a` is divided by `b`.
- It is acceptable for the last item in the output to be appear twice in your Scheme program even if it appears only once in Python.

3. [5 marks]

Write a Scheme function `ack` that takes two integers `i` and `j` as parameters and returns $A(i, j)$, the value of Ackermann's function. This function is defined for $i, j \geq 1$; here it is:

$$A(1, j) = 2^j \text{ for } j \geq 1$$

$$A(i, 1) = A(i - 1, 2) \text{ for } i \geq 2$$

$$A(i, j) = A(i - 1, A(i, j - 1)) \text{ for } i, j \geq 2$$