Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Sandeep Sadanandan

Sommer Semester 2009
Problem Sheet 8
July 5, 2009

Python For Fine Programmers

Deadline: July 9, 2009

Problem 1 (4 Points)

Create a class where (without using another class with `@classmethod`) the `getInstance` returns always the same objeck. In essence, the class can have no more than a single instance at any given time. The `id` of both the objects is the same in the code below.

```
>>> if __name__ == "__main__":
...     obj1 = SingleObjects('object1')
...     print id(obj1), obj1
...
...     obj2 = SingleObjects('object2')
...     print id(obj2), obj2

Creating object1
47685972607376 <__main__.SingleObjects object at 0x2b5ec1754190>

Creating object2
47685972607376 <__main__.SingleObjects object at 0x2b5ec1754190>
```

Problem 2 (4 Points)

Write a tracer function/class for recursive functions. The tracer is to be implemented with decorators. The tracer has to keep track of the call and return of the function calls.

Every call of function has to be printed with the given parameter and every return from the function also has to be represented with the return result. The decorator also has to represent the depth of the recursive call by intending the trace-output accordingly.

```
@mtrace('fib')
def fib(n):
    return n if n < 2 else fib(n - 1) + fib(n - 2)

@mtrace('fact')
def fact(n):
    return 1 if n == 0 else n * fact(n-1)

print "\n--- Fib(3) ---\n", fib(3)
```

```
10 print "\n--- Factorial(4) ---\n", fact(4)
11 [sadanand@lxmayr10 @ ~]python goooooooooo.py
12
13 --- Fib(3) ---
14   Call[1]: fib(3,)
15    Call[2]: fib(2,)
16     Call[3]: fib(1,)
17     Result[3] : 1
18     Call[3]: fib(0,)
19     Result[3] : 0
20    Result[2] : 1
21    Call[2]: fib(1,)
22    Result[2] : 1
23   Result[1] : 2
24 2
25
26 --- Factorial(4) ---
27   Call[1]: fact(4,)
28    Call[2]: fact(3,)
29     Call[3]: fact(2,)
30      Call[4]: fact(1,)
31       Call[5]: fact(0,)
32       Result[5] : 1
33      Result[4] : 1
34     Result[3] : 2
35    Result[2] : 6
36   Result[1] : 24
37 24
```

Problem 3 (4 Points)

Using the server-client built using sockets and using the graph isomorphism and hamiltonean path checking methods, implement a crude zero-knowledge-proof system.

Problem 4 (2 Points)

Implement a personal version of dictionary in python where the items are always sorted.