University of Bath

DEPARTMENT OF COMPUTER SCIENCE EXAMINATION

CM10227: PROGRAMMING 1

Thursday, 23 January 2014, 16.30–18.30

No calculators may be brought in or used.

Full marks will be given for correct answers to THREE questions. If you opt to answer more than the specified number of questions, you should clearly identify which of your answers you wish to have marked. In cases where you have failed to identify the correct number of answers the marker is only obliged to consider the answers in the order they appear up to the number of answers required.

PLEASE FILL IN THE DETAILS ON THE FRONT OF YOUR ANSWER BOOK/COVER AND SIGN IN THE SECTION ON THE RIGHT OF YOUR ANSWER BOOK/COVER, PEEL AWAY ADHESIVE STRIP AND SEAL.

TAKE CARE TO ENTER THE CORRECT CANDIDATE NUMBER AS DETAILED ON YOUR DESK LABEL.

DO NOT TURN OVER YOUR QUESTION PAPER UNTIL INSTRUCTED TO BY THE CHIEF INVIGILATOR.

1. (a) Write a recursive and iterative function/method that implements the behaviour of the *modulo* operator for two positive integers. You are not allowed to use the one provided by the language nor are you allowed to use division. One version should be implemented in Python, the other one in Java. [10]

   (b) What is *garbage collection*? [2]

   (c) What are *mutable* data types in Python? How do they behave differently from *immutable* data types when passed as parameters? [4]

   (d) Explain *aliasing* of variables. Provide a Java example of variables that are aliases and provide an example where two variables are referring to the same value but are not aliases. [4]

2. (a) Consider the following piece of code:

```
# this function takes a list of numbers
# and three functions as arguments.
def badFunc (numbers, func1, func2, func3):
        # It will apply an inner function f to each element
        # f returns the application of a function on its parameter.
        # The function applies func1 if the element is divisible by 3
        # If not, it applies func2 if the number is even
        # or func3 if the number is odd and larger than 8
        # otherwise (as a default) it returns -1
        def f (x):
                if (x / 3 == 0):
                        return func1(x)
                if (x / 2 == 0):
                        return func2(x)
                for el in numbers:
                        el = f(el)
```

While the comments are correct, something went wrong with the code. When we run:

```
# correct code
def func1(x): return 100
def func2(x): return x/2
def func3(x): return x*3

numbers1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print numbers1

badFunc(numbers1, func1, func2, func3)
print  numbers1
```

we expected to see:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, -1, 100, 12, 15, 100, 21, 24,  100]
```

Instead we got:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Remove the logical error(s) and explain what went wrong.

[10]

(b) We discussed two ways of differentiating programming languages on the basis of data types, *statically* vs. *dynamically* typed and *weakly* vs. *strongly* typed. Explain each term (static, dynamic, weak and strong) and mention in which categories you place Python and Java. [4]

(c) What is the difference between *syntax* and *run-time* errors (bugs)? Name one other kind of bug. [3]

(d) Explain the concept of *polymorphism* in the context of the Java programming language. [3]

3. (a) Consider the following piece of code:

```
def recList(list1, list2=[], list3=[2,4,6]):
        if len(list1)=0:
                print list 2*3
                return
        el=0
        t=0
        for i in range (len(list1)):
                list2.append(list1.pop(i)*list3[el]+i)
                el=(el+1)%len(list3)
                t=t+recList(list1, list2)
                list1.insert(i, list2.pop())
        return t

recList([2,4],[3,5,1])
```

What is printed on the screen? What is the return value of the function call? [10]

(b) Why is *data encapsulation* important? [3]

(c) Explain the concept of a *constructor* in the context of Java classes. [3]

(d) Explain the concept of *inheritance*. How is this implemented in Java? [4]

4. (a) Explain the difference between *method overloading* and *method overriding.* [4]

(b) Describe two key components of *incremental programming.* [2]

(c) Explain the concepts of *abstraction* and *modularisation.* [4]

(d) A deque ("double-ended queue") is like a queue, except that it allows access at both ends. Create a Java class Deque which implements a deque with the capacity to hold 100 items. It should implement the following interface:

```
public interface DequeInterface

        public void addFront(Object o);
                //adds item o to the front of the deque

        public void addRear(Object o);
                //adds item o{\em  to the rear of the deque

        public Object removeFront();
                // removes the item at the front and returns it

        public Object removeRear();
                // removes the item at the rear and returns it
```

[10]

5. (a) Why is *code robustness* considered desirable for a program? [2]

(b) List two contexts in which a programmer would need to rely on strong *documentation.* [2]

(c) Explain the following statement: Java is not completely *object-oriented.* Give examples to support your explanation. [5]

(d) Design a Java program which reflects the roles and the hierarchy of bakery school members as follows:

For each of the following roles, create a class/interface/abstract class (as appropriate): school member, apprentice, baker, seller, owner.

- Everyone is a school member.
- Owners as well as bakers are sellers.
- Apprentice is a subclass of baker.

Add to the appropriate classes data fields to capture the following information:

- Every school member has a name.
- Every baker and apprentice has a specialism.
- Every seller has a primary client.
- Owners (and, in this case, only owners) teach courses.

Add methods for getting and setting the content of these fields. You must provide an interface for each method. A more detailed implementation of the classes is not required.

[11]