This paper contains questions designed to test both your understanding of core programming concepts and your ability to demonstrate that understanding whilst writing short programs (or parts of programs). For the sake of clarity, you should note that the Java language should be used whenever a question asks you to either write code or provide examples. On that basis, wherever the paper asks you to provide one or more programs, classes, methods, statements, expressions, fields, etc. your answers should conform to the syntax rules of the Java language as taught in this year's CM10227 lectures (i.e. Java version 7).

1. (a) Write a *recursive* and *iterative* function in C that prints out the first 10 Fibonacci numbers. [10]

```
recursive version:
3 marks for working recursion.
1 mark for correctly implemented base case
1 mark for using printf.

iterative version:
3 marks for working iteration.
1 mark for correctly calculated/implemented number of iterations
1 mark for using printf
```

(b) In C describe the effects of declaring an char array. Give and example of declaring an car array. Give 3 examples of initialising a char array. [6]

```
2 marks for it reserves an area in memory for
  the declared number of chars
1 mark for char[3] my_char_array;
1 mark for char[3] my_char_array = {'a','b','c'};
1 mark for char[3] my_char_array = "Hi";
1 mark for char[3] my_char_array; my_char_array[0] = 'a';
```

(c) What is the difference between a Statically typed and Dynamically typed programming language? [2]

```
1 mark for Statically typed - A language in which types are
  fixed at compile time
1 mark for Dynamically typed - A language in which types are
  discovered at execution time
```

(d) What is Encapsulation and Generalisation? [2]

```
1 mark for encapsulation is wrapping a piece of code in a function
1 mark for
taking something specific and making it more general
```

2.

(a) What is the definition of an Abstract Data Type? [2]

```
1 mark for Data Structure
1 mark for functions that operate on the Data Structure
```

(b) Write a Abstract Data Type in C for a Stack that uses an array to store a maximum of 100 integers. [10]

```
1 mark for a Stack struct
1 mark for int max size
1 mark for int array[100]
1 mark for head
2 marks for push function
2 marks for pop function
2 marks for peek function
```

(c) Name three different kinds of *programming error*. Give an example of each one. [6]

```
1 mark for each type, 1 for each example
Syntax Error:
e.g. (5+2))
Logical Error
e.g. Incorrectly written formula
Run Time Error
e.g. Divide by zero
```

(d) What is the definition of a Tuple? [2]

```
1 mark ordered collection of values
1 mark of different types
```

3.

(a) Explain the execution of this Java program and what its output is. [10]

```
1. public class AproxSquareRoot{
2.
3.     public static void main(String[] args){
4.         AproxSquareRoot asr = new AproxSquareRoot();
5.         System.out.println(asr.calculateFor(125, 5));
6.     }
7.
8.     public double calculateFor(int number, int accuracy){
9.         String doubleAsString = Integer.toString(number);
10.        int doubleLength = doubleAsString.length();
11.        double x = number * doubleLength;
12.        int i = 0;
13.        while(i <= accuracy){
14.            x = 0.5 * (x + (number/x));
15.            i++;
16.        }
17.    return x;
18.    }
19. }
```

2 marks for defining a class called AproxSquareRoot which is instantiated in the main method
1 mark for method calculateFor called in main method and result printed to standard out
1 mark for calculateFor takes two parameters number and accuracy
1 mark for line 9 using the Integer Java API to convert number to a String
1 mark for line 10 using the String Java API to find the length of the String in line 9
2 marks for while loop, looping from 0 to accuracy and performing the calculation on line 14

2 marks for out put is 11.6830

(b) Explain the difference between Call-by-Reference and Call-by-Value when passing arguments to methods in Java. [6]

Call–by–Reference
2 marks for the caller gives the called method the ability to directly access to the callers data

1 mark for caller can modify that data

Call–by–Value
2 marks for a copy of the actual parameter is passed to the formal parameter of the called method
1 mark for any change made to the formal parameter will have no effect on the actual parameter

(c) What are the advantages of Inheritance? [2]

2 marks from any of the following...
avoids code duplication
allows code reuse
simplifies the code
simplifies maintenance and extending

(d) What are the advantages high-level programming languages? [2]

2 marks for any of...
Easier to program
Less time to write
Easier for people to read
Easier for people to correct
Much easier to port, or modify to run on different computers

4.

(a) ... [10]

(b) Describe the Dimond Problem. How does Java's inheritance rules address this problem? [6]

4 marks for description...
(i) Two classes B and C inherit from A and
class D inherits from both B and C.
(ii) If there is a method in A that B and/or C has overridden,
and D does not override it then which version of the
method does D inherit: that of B, or that of C?

1 mark for Java does not allow multiple inheritance for classes
1 mark for Java does allow programmers to implement multiple Interfaces

(c) How are the inheritance rules different for *interfaces* and *superclasses* in Java? [2]

subclasses can inherit from multiple interfaces (1)
but only one superclass (1)

(d) What is the difference between a Strongly typed and Weakly typed programming language? [2]

1 mark for Strongly typed – A language in which types are always enforced
1 mark for Weakly typed – A language in which types may be ignored

5. (a) Explain the execution of this C program and what its output is. [10]

```
1. #include <stdio.h>
2. #include <math.h>
3.
4. double aprox_pi(int term){
5.     double pi = 1.0;
6.     double denominator = 3.0;
7.     int is_addition = 0;
8.     int i;
9.     for(i=0; i<=term; i++){
10.        if(is_addition){
11.            pi = pi + (1.0/denominator);
12.            is_addition = 0;
13.        }
14.        else{
15.            pi = pi - (1.0/denominator);
16             is_addition = 1;
17.        }
18.        denominator = denominator + 2.0;
19.    }
20.    return (4*pi);
21. }
22.
23. int main(void) {
24.    double pi = aprox_pi(5);
25.    printf("%1.3f\n",pi);
26.    return 0;
27. }
```

2 marks main defines a double pi which is set to the return value of aprox_pi
2 marks for aprox_pi initialising variables and looping from 0 to term
1 mark if statement determining whether to + or − 1/denominator
1 mark setting is_addition to 0 or 1 which is like a boolean
1 mark denominator incrementing by 2.0 on each loop
1 mark for returning 4*pi

2 marks for output 3.284 to 3 decimal places as specified by the format specifier

(b) In Java ArrayLists, Stacks and arrays have different ways to add and remove elements. Describe these differences with examples. [6]

ArrayLists
2 marks for ArrayLists using methods add and remove(index)
Stacks
2 marks for Stacks pushing and popping
arrays
2 marks for arrays adding via and index to an element and removing by setting the element to NULL

(c) What are the advantages low level programming languages? [2]

1 mark for program can be tuned to a specific computer
1 mark for above allows for maximum execution speed and minimum memory consumption.

(d) What is the difference between an *argument* and a *parameter*? [2]

a parameter is what appears in the definition of a method (1)
an argument is the instance passed at runtime (1)