The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 1 MODULE, SPRING SEMESTER 2018-2019

PROGRAMMING PARADIGMS

Time allowed: **TWO Hours THIRTY Minutes**

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer ALL questions.
(Each question is worth equal marks)

Only silent, self-contained calculators with a Single-Line Display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

ADDITIONAL MATERIAL: Haskell standard prelude.

INFORMATION FOR INVIGILATORS: Exam papers must be collected at the end of the exam.

1. Object-Oriented Programming/Java (25 marks)

(a) Name and briefly describe 4 primitive numeric data types.

(4)

(b) Fill in the missing lines ?? (before the comments) that are required to complete the following program and give the desired output of:

```
1 2 3 4 5
H E L L 0

public class Exam1 {
    public static < E > void printArray( E[] inputArray ) {
        for(E element : inputArray) {
            System.out.printf("%s ", element);
        }
        System.out.println();
    }

    public static void main(String args[]) {
        ??                          //Define an integer array
        ??                          //Define a character array
        ??                          //Pass integer array to Method
        ??                          //Pass character array to Method

    }
}
```

(4)

(c) What key feature of Object Orientation does the program in the previous question exemplify and why.

(3)

(d) Describe, with examples, 3 different types of polymorphism.

(6)

(e) Briefly, what is method overriding and method overloading and whether they are they compile or runtime concepts.

(3)

(f) Discuss briefly why Java is generally seen as a safer language than C.

(3)

(g) What is the difference between an inner class and a sub-class.

(2)

2. Object-Oriented Programming/Java (25 marks)

(a) What is Java bytecode. Discuss a possible advantage and possible disadvantage that this gives Java over traditional programming language implementations.

(3)

(b) What is the output from the following code.

```
public class Test2{
	public static void main(String args[]){
		int n;
		try{
			n =  calc ();
			System.out.println(n);
		}catch(Exception e) {
			System.out.println("Error");
		}
				try{
		n =  calculate ();
		System.out.println(n);
		}catch(Exception e) {
			System.out.println("Error");
		}

	}
	static int calculate(){
		return (2/3);
		}
	static int calc(){
		return (2/0);

	}
}
```

(4)

(c) What is the 'final' keyword used for in Java, give examples of its use.

(6)

(d) The following code fails to compile, what operator is missing and what will the output be once the operator is correctly inserted, compiled and executed.

```
public class Calculate {
	int num = 10;
	public void calc(int num) {
		this.num = num * 8;
	}
public void printNum(){
```

```
        System.out.println(num);
        }
    public static void main(String[] args) {
        Calculate obj = Calculate ();
        obj.calc(2);
        obj.printNum();
        }
    }
```

(4)

(e) Examine the following code, write the 4 lines of output, giving the missing text (represented by ??):

```
class People {
        public int age;
        People(int age) {this.age = age;}
}

public class TestPeople {
        static void change1(People p) {
                System.out.println("(1)" + p + " age: " + p.age);
                People newP = new People(p.age);
                p = newP;
                System.out.println("(2)" + p +" age: " + p.age);
        }
        static void change2(People p) {
                System.out.println("(3)" + p + " age: " + p.age);
                p.age += 10;
                System.out.println("(4)" + p + " age: " + p.age);
        }
                public static void main(String[] args) {
                People p = new People(25);
                change2(p);
                change1(p);
        }
}
```

Output:

```
??????????@12ab9876 ???????
???????????????? ???????
???????????????? ???????
??????????@cd456789 ???????
```

(4)

(f) Discuss the main concept the previous question highlights

(4)

3. Functional Programming / Haskell (25 Marks)

(a) Write down *the most general types* for the following functions:

(i) one x = [x]

(1)

(ii) two x = (x,x)

(1)

(iii) first x y = x

(1)

(iv) f = id

(1)

(v) twice f x = f (f x)

(1)

(b) Fill in the missing parts ? in the following definitions to make them type correct. It does not matter what they do as long as they are type correct:

(i)
```
e1 :: Int -> (Int -> Int)
e1 = \x -> ?
```

(1)

(ii)
```
e2 :: ?
e2 = [1,2,3]
```

(1)

(iii)
```
e3 :: a -> b -> (a,b,a,b)
e3 ? = ?
```

(1)

(iv)
```
e4 :: [a] -> [a] -> [a]
e4 ? = ?
```

(1)

(v)
```
e5 :: ?
e5 = (True,False,True)
```

(1)

(c) Define the following library functions using explicit recursion (without the use of fold):

(i) reverse :: [a] -> [a]

(2)

(ii) map :: (a -> b) -> [a] -> [b]

(2)

(iii) product :: [Int] -> Int

(2)

(iv) concat :: [[a]] -> [a]

(2)

(v) replicate :: Int -> a -> [a]

(2)

(d) For the remainder of this question, suppose that arithmetic expressions built up from integer values using subtraction are represented as follows:

```
data Expr = Val Int
          | Sub Expr Expr
```

(i) Write a Haskell expression using the Expr type which represents the arithmetic expression (2 - 6) - 10.

(1)

(ii) Define a function eval : Expr -> Int which evaluates Expr expressions. For example, your answer to (i) should evaluate to -14

(2)

(iii) Define a function subs :: Expr -> Int that counts the number of subtractions in an Expr expression. For example, your answer to (i) should evaluate to 2.

(2)

4. Functional Programming / Haskell (25 Marks)

(a) Explain what are curried functions and why they are used and preferred over uncurried functions? To what other concept are they related which gives a formal meaning to curried functions? Give a short explanation of this concept with syntax and examples.

(5)

(b) What is a *higher-order function*? Give two reasons why higher-order functions are useful, illustrating your reasons by simple examples.

(3)

(c) What is lazy evaluation in Haskell and how does the compiler achieve it?

(2)

(d) For the remainder of this question, suppose that you have a key-value store defined using the recursive data definition like this:

```
data KVStore k v = KVPair k v ( KVStore k v)
                 | Empty
```

Such a key-value store allows the storage of key-value pairs where the order is not important.

(i) Define an expression of type KVStore Int String which contains the following key-value pairs: 1 -> "Chris", 2 -> "Jonathan".

(2)

(ii) Write a function kvsLookup :: ? -> Maybe ?, that searches the value v to a key k and returns a Maybe as result. First define the type of kvsLookup by replacing the ? with the correct types and then implement the function.

(5)

(iii) Write a function that deletes a value *v* to a given key *k* from the key-value store. If the key does not exist in the key-value store, then this function has no effect on the key-value store.

(5)

(iv) Write a function that prints a key-value store to the console using interactive IO (). The format should be "KEY: VALUE" in a line for each key-value pair. Hint: you might need the 'Show' typeclass constraint.

(3)