SEASON 2 EXAMINATION 2016 - 2017

AUTOMATED CODE GENERATION

DURATION 90 MINS (1.5 Hours)

This paper contains 5 questions

Answer all of Section A and choose THREE questions from Section B. Each question is worth 25 marks and there are 100 marks available in total. You should aim to spend around 22.5 minutes on each question.

An outline marking scheme is shown in brackets to the right of each question.

This examination is worth 60% of the overall module mark. The coursework is worth 40%.

University approved calculators MAY be used.

A foreign language dictionary is permitted ONLY IF it is a paper version of a direct Word to Word translation dictionary AND it contains no notes, additions or annotations.

7 page examination paper.
Section A

Question A1.

(a) List three reasons why code reuse based on libraries and components has reached its limits and suggest one way in which code generation may help this situation.

[5 marks]

(b) In the architecture of a typical code generator, a transformation database contains AST rewrite rules to be used as input to the generator engine. What is the advantage of doing this as opposed to coding these rewrites in the engine itself?

[5 marks]

(c) Explain the difference between generative and transformative code generation. Give an example of a generative code generation technique.

[5 marks]

(d) Explain the difference between a function and a macro in C.

[5 marks]

(e) Explain what is meant by the term *quote-based* metaprogramming.

[5 marks]
Section B

Question B1.

(a) Explain the process of *macro expansion* in C. [8 marks]

(b) To initialise a variable of compound struct type in C we can assign the variable to `{0}`. For example,

```c
struct Link {
    double value;
    struct Link* next;
};
Link list = { 0 };
```

declares a variable list and initialises the structure to a null value. After using list, to reset it to its initial null state efficiently we can copy the contents of a constant “Zero” instance of the Link structure initialised to the null value into list. Function `memcpy` that accepts three arguments, a pointer to a destination structure, a pointer to a source structure, and the number of bytes N to copy is useful for this.

Write C macros `INIT`, `INITZERO`, `RESET` that can be used, for any compound type, to

- (i) declare variables of compound type that are initialised to `{0}`
- (ii) declare and initialise a “Zero” instance of a compound type
- (iii) reset variables of compound type

respectively. You may find the function `sizeof` that returns (in bytes) the size of a given structure useful.

[11 marks]

(c) C Macros suffer from hygiene problems due to incorrect use. Describe three ways in which your macros from the previous part of the question may be used incorrectly.

[6 marks]
Question B2.

(a) Explain the difference between a joinpoint and a pointcut descriptor.  
[4 marks]

(b) Explain what is meant by an inter-type declaration in AspectJ.  
[4 marks]

(c) Describe in words the methods matched by the following signature patterns:

(i) * void java.awt.*.set*(..)  
(ii) !public * java..*.*Listener(*)  

[6 marks]

(d) Consider a Java class that implements the interface

```java
interface SensorInterface {
    public double getClock();
    public double getTemp();
}
```

The class represents temperature sensors where the two methods return the time elapsed (in milliseconds) since construction and the current sensor temperature reading respectively.

Write an aspect in AspectJ so that if the `getTemp` method is called on any instance of a class that implements `SensorInterface` within one second of a value being returned from a previous call on the same instance then this previous value is returned immediately instead of actually executing the `getTemp` method. You may use the sensors’ own internal clocks to determine the time elapsed between calls.  
[11 marks]
Question B3.

(a) Explain the quotation/unquotation mechanism in Meta-AspectJ.

[6 marks]

(b) Explain the semantics of the infer keyword in Meta-AspectJ.

[4 marks]

(c) A copy constructor is a Java constructor method that has the form

```java
public SomeClass(T1 f1, T2 f2, ... , TN fN){
    this.f1 = f1;
    this.f2 = f2;
    ...
    this.fN = fN;
}
```

Write a Meta-AspectJ class that contains a method called

```java
String generateCopyConstructor(String cn, String[] flds)
```

that accepts the String name of a class and an array containing String names of fields. The method should return Java code for a copy constructor method for the given class that accepts an argument for each field listed in flds.

You may assume that there exists a method

```java
String getFieldType(String cn, String fn)
```

in your class that returns the String name of the type of field fn in class cn if it exists, otherwise a NoSuchFieldException is thrown. Recall that the entry points for statements and formal declarations in Meta-AspectJ are Stmt and FormalDec respectively. [15 marks]
Question B4.

(a) Describe which compile time errors may arise during C++ template instantiation. Your answer should make specific reference to the template specialisation model.

[6 marks]

(b) Consider the following templated class and its specialisations:

a) template<class T, int N, int M>
   class C { ... }

b) template <class T, int N>
   class C<T,N,0> { ... }

c) template<int N, int M>
   class C<int,N,M> { ... }

d) template<>
   class C<bool,0,0> { ... }

for each of the following, say which template specialisation, if any, would be selected by the compiler and why.

(i) C<int, 0, 1>
(ii) C<char,1,0>
(iii) C<int,0,0>
(iv) C<bool,0,0>
(v) C<char,1,1>

[6 marks]

(c) Consider the following templates that represent a List data structure.

```cpp
struct Nil
{   enum { head = ERROR };
    typedef Nil Tail;
};
```
template <int head_, class Tail_ = Nil>
struct Cons
{
  enum { head = head_);
    typedef Tail_ Tail;
};

Write a meta-function using templated structs that accepts such a List structure and statically duplicates each element of the list. For example, the duplicate of [4; 3; 7; 1] is [4; 4; 3; 3; 7; 7; 1; 1].

[13 marks]