Answer four out of five questions.

Each question is worth 15 marks.
This examination is worth 60% of the overall module mark.
The coursework is worth 40% of the overall module mark.

Calculators may NOT be used.
1. a) Define "meta-program". 
   (2 marks)

   b) Explain the difference between a function and a macro. 
   (4 marks)

   c) List three non-code artefacts that a code generator can provide. 
   (3 marks)

   d) List three reasons why reuse based on libraries and components has reached its limits. Explain how code generation techniques can overcome these limitations. 
   (6 marks)

2. a) Explain the quotation / unquotation mechanism used in Meta-AspectJ. 
   (6 marks)

   b) Write a Meta-AspectJ method that takes as input a class name and the syntactic representation of an arbitrary logging statement, and produces as output a new AspectJ aspect that counts all calls to methods in the given class, and executes the given logging statement when each method is called. 
   (9 marks)
3. COMP6035W1

3. a) Explain the meaning of joinpoint in AspectJ, and the difference between static and dynamic joinpoints.

(3 marks)

b) Describe in words which joinpoints are matched by the following pointcut descriptors, and describe for each pointcut descriptor whether it describes static or dynamic joinpoints.

(i) \text{call}(\text{int} c.\text{meth}(\text{int})) \&\& \text{args}(1)
(ii) \text{cflowbelow}(\text{call}(\text{int} c.\text{meth}(\text{int})))
(iii) \text{execution}(\text{int} c.\text{meth}(\text{int}))

(6 marks)

c) Consider the following AspectJ program below:

```java
class A {
    void m(int i) { ...};
}

class B extends A {};

class Test {
    void test() {
        A a = new B();
        B b = new B();
        a.m(1);
        b.m(2);
    }
}
```

For each of the pointcut descriptors below, describe which joinpoints are matched.

(i) \text{call}(\text{* A+.m(\ldots)})
(ii) \text{execution}(\text{* B.m(\ldots)})
(iii) \text{execution}(\text{* A.m(\ldots)}) \&\& \text{target}(B)

(6 marks)
4. a) Explain the use of the special proceed( .. ) command when executing around advice in AspectJ.  

(3 marks)

Java applications use the java.io.PrintStream System.out object to output data to the standard output device. All character data sent to this object by a variety of print and write methods will be encoded using a character encoding such as UTF-8. A character encoding is simply a mapping from character values to byte values.

For input, Java uses the java.io.InputStream System.in object. This supports a method

\[ \text{int read(byte[]} b) \]

that reads a number of bytes from the input and stores them in a byte array b. The number of bytes read is returned as an int.

For the purposes of this question you may ignore all import statements and all exception handling related to input/output.

b) Write an aspect ChangeEnc that ensures that all output written to System.out in a Java application is output using the UTF-16 encoding.

Hint: You will find this call to the PrintStream constructor useful:

\[
\text{PrintStream myOut = new PrintStream(System.out, true, "UTF-16");}
\]

This creates a new PrintStream object that uses UTF-16 encoding to output to the standard output device.

(6 marks)
5. COIVIP6035W1

c) Extend your aspect ChangeEnc so that it also ensures that any input read using the above read method is decoded from UTF-16 before it is used in the Java application.

Hint: You will find the following code useful:

```java
Charset enc = new Charset("UTF-16", null);
void translate(byte[] b) {
    ByteBuffer bb = ByteBuffer.wrap(b);
    byte[] eb =
        enc.decode(bb).toString().getBytes();
    System.arraycopy(eb, 0, b, 0, eb.length);
}
```

The translate method converts a byte array in "UTF-16" encoding to a byte array in the default system encoding.

(6 marks)

5. a) Consider the six template definitions below:

(I) template <class S, class T>
class P {...};

(II) template <class S, class T>
class P<S*, T> {...};

(III) template <class S, class T>
class P<S, T*> {...};

(IV) template <class T>
class P<T, T> {...};

(V) template <class T>
class P<T*, T*> {...};

(VI) template <>
class P<int*, int*> {...};

TURN OVER
6. COMP6035W1

For each of the following five declarations describe which specialization is selected, and why, or describe why the declaration is causing an error.

(i) P<bool*, int> a;
(ii) P<bool, bool> b;
(iii) P<bool, int> c;
(iv) P<int*, bool*> d;
(v) P<int*, int*> e;

(5 marks)

b) Consider the class MyVector given below. Write a template metaprogram that generates a specialized instance of the mean average of an element of the class MyVector. The mean average of a vector x of length N is defined as m = \( \sum x[i] / N \).

```cpp
template<class T, int N>
class MyVector {
    public:
        T operator[](int i) const
        { return data[i]; }

    private:
        T data[N];
};
```

(10 marks)

END OF PAPER