5. **Exceptions and Generics**

(a) In Java, erroneous behaviour often results in an Exception being generated. An Exception allows a programmer an opportunity to handle an error gracefully and possibly take corrective measures.

i. Performing a division by zero in Java causes an exception when dealing with integer division. However, for floating point numbers, the special value "`Inf`" is returned and no exception is raised. Write a *checked* exception class that will be raised when a floating point division by zero is about to occur. [3]

ii. Write a method that divides one floating point number by another and returns the result. In the event that the denominator is zero, an exception should be generated. [4]

iii. With a code example, demonstrate how you would use your division method in a Java application. [3]

(b) In Java, *Generics* can be used to provide type information at compile-time.

i. Explain two of the motivating factors behind using Generics in Java. [2]

ii. Design a `KeyValuePair` class where the value **must** be a numerical value. [6]

iii. Show how you would initialise a `KeyValuePair` object in Java. [2]