1. a.
    - Functional programming is declarative and focuses on “what” while imperative programming focuses on “how”.
    - Functional programming has no side affect since (state does not change) while imperative programming has side effects since it changes states via access to its environment variables.

    b. Declarative programming focuses on what needs to be done. It uses functions to control its flow and has no side effects and less prune to errors. As an example, SQL uses declarative principles to get data from the database

    c.

Definition	Example
1. A functional interface is an interface with only one abstract method.	Comparator<T> Interface
2. A Functor is an implementation of a functional Interface	Employee implements Comparator<Employee>
3. A closure is a block of code that behaves like a function and makes use of its environment variables (class or method variables)	A method in Employee that has access to the instance variables of Employee.

    d.

    I. Thread-safe
    II. Has no side-effect
    III. Easier to write, read and understand due to its compact nature.

    e.

    I. λx.x + 2 * x * x
    II. λxy.y – x + Math.pow(x,y)
    III. λxyz.z – (x + y)

    f.

Number	Parameter	Free variables
i.		s, t
ii.	u, v	a, b, x
iii.	s, t	

    g. interface Predicate<T>

    h.

    I. x -> p.println(x)
    II. Object::instanceMethod references because out is an object with the println method.