"Turing machines and the lambda calculus are equivalent in computational power: each can efficiently simulate the other."

"lambda calculus and Turing machines not just closely related but they are **equivalent models of computation**."

"Turing machines and Lambda Calculus are two models that capture the notion of algorithm (mechanical computation). Lambda calculus was invented by Church to perform computations with functions. It is the basis of functional programming languages. Basically, every problem that is computable (decidable) by Turing machines is also computable using Lambda calculus. So, they are two equivalent models of computation (up to polynomial factors) and both try to capture the power of any mechanical computation."

lambda calculus and Turing machines both compute the same class of number-theoretic functions, they are not precisely equivalent in every way imaginable. For example, in realizability theory there are statements which can be realized by a Turing machine but not by lambda calculus. One such statement is the formal Church's thesis, which states:

â^€f:natâ†'nat â^ƒe â^€n â^ƒk (T(e,n,k)â^§U(k,f(n)))â^€f:natâ†'nat â^ƒe â^€n â^ƒk (T(e,n,k)â^§U(k,f(n)))

"Here T is Kleene's T predicate. A realizer for this statement would be a program cc that accepts a (representation of) map f and outputs (a representation of) ee with the desired property. In the Turing machine model the map ff is represented by the code of a Turing machine that computes f, so the program cc is just (the code of a Turing machine computing) the identity function. However, if we use the lambda calculus, then cc is supposed to compute a numeral representing a Turing machine out of a lambda term representing a function f. This cannot be done"

ADVANTAGE OF TURING MACHINE

"Turing machines are similar to finite automata/finite state **machines** but have the **advantage** of unlimited memory. They are capable of simulating common computers; a problem that a common computer can solve (given enough memory) will also be solvable using a **Turing machine**, and vice versa"

DISADVANTAGE

"A limitation of **Turing machines** is that they do not model the strengths of a particular arrangement well. ïµ b. For instance, modern stored-program computers are actually instances of a more specific form of abstract **machine** known as the random access stored program."

ADVANTAGE OF LAMBDA CALCULUS

"The **benefit of lambda calculus** is that it's an extremely simple model of computation that is equivalent to a Turing machine. But while a Turing machine is more like assembly language, **lambda calculus** is more a like a high-level language."

DISADVANTAGE OF LAMBDA CALCULUS

"The **lambda calculus** does not specify an evaluation order. No special forms (this is related to the last bullet). Special forms in Scheme are just expressions that are evaluated using a different evaluation order than the default. As mentioned, the evaluation order is irrelevant in the **lambda calculus**."