1.

a l-value of a variable is its address, r-value of a variable is its value.

b. static: Bound to memory cells before execution begins and remains bound to the same memory cell through execution. Ad: efficiency, because it can be accessed directly. Dis: lack of flexibility.

stack-dynamic: Storage bindings are created for variables when their declaration statements are elaborated. Ad: allows recursion, conserves storage. Dis: Overhead of allocation and deallocation; subprograms cannot be history sensitive; indirect addressing.

Explicit heap-dynamic: Allocated and deallocated by explicit directives, specified by the programmer, which take effect during execution. Ad: Provides for dynamic storage management. Dis: inefficient and unreliable.

Implicit heap-dynamic: Allocation and deallocation caused by assignment statements. Ad: flexibility. Dis: Inefficient, because all attributes are dynamic, and loss of error detection.

c. Advantage of Dynamic scoping: convenience, since we do not need to care about the type of variables. Disadvantages: while subprogram is executing, its variables are visible to all subprograms it calls. Impossible to statically type check. Poor readability, since it is not possible to statically determine the type of variable.

2.

a. Strongly typed: Strongly typed means that type errors are always detected in a programming language.

b. Java is a strongly typed language, because whenever we declare a variable, we have to specify what type is the variable. Variable can not be declared without knowing the range of values it can hold.

3.

a. Let row, col, depth be the length of each rows, columns, depths.

Row major:

Location(a[a, b, c]) = starter address + c + (a * col + b) * depth

Column major:

Location(a[a, b, c]) = starter address + c + (b * row + a) * depth

b. For: java uses garbage collection, so implicit heap story recovery is done by the run-time system automatically. Against: but recovering the heap storage memory implicitly is hard for programmers to understand, and when they are programming, it is easy to make mistakes.

c. Since in c#, pointers only store the memory address of value types and arrays. It does not tracked by garbage collection. When it is not absolutely necessary, it is not much used.

d. They should be both used in programming languages. Because they have different features. Jagged matrices help in managing the memory space, since they can have different length of array which can handle different situations without wasting memory(having a lot of nulls). Rectangular matrices is easier to work with since it has same length in rows.