CS 336 – Assignment 6

Question 1

When execution reaches position 1 inside function **b**, the call sequence is:

main → bigsub() → a() → c() → b()

In this program, **bigsub** is at level 1, **a** is at level 2, and both **b** and **c** are at level 3.

At position 1, the stack contains the following activation record instances (from top to bottom):

Activation Record	Static Link →	Dynamic Link →
b	a	c
c	a	a
a	bigsub	bigsub
bigsub	main (or null)	main (or null)
main	—	—

The **dynamic chain** (also called the call chain) is **b → c → a → bigsub → main**.
The **static chain** is **b → a → bigsub → main**.

The dynamic link connects to the activation record of the caller, while the static link connects to the activation record of the lexically enclosing function.
At this moment, the environment pointer refers to the base of **b**'s activation record.

Question 2

When execution reaches position 1 inside function **d**, the call sequence is:

main → bigsub() → a(true) → b() → a(false) → c() → d()

Here, **bigsub** is level 1, **a** is level 2, **b** and **c** are level 3, and **d** is level 4.

At position 1, the stack contains the following activation record instances (from top to bottom):

Activation Record	**Static Link →**	**Dynamic Link →**
d	c	c
c	bigsub	a(false)
a(false)	bigsub	b
b	a(true)	a(true)
a(true)	bigsub	bigsub
bigsub	main (or null)	main (or null)
main	—	—

The **dynamic chain** is **d → c → a(false) → b → a(true) → bigsub → main**,
and the **static chain** is **d → c → bigsub → main**.

The static link connects each activation record to its static parent based on lexical scope, and the dynamic link connects to the subprogram that called it.
At this point, **d**'s activation record is at the top of the stack, and its environment pointer indicates that location.

Question 3

If every activation record were given two static links—one to its immediate static parent and another to its static grandparent—this would slightly change both linkage time and non-local reference time.

During a subprogram call, the system would need to assign both static links, which would make subprogram linkage a little slower.
However, this cost is constant and very small because only one extra pointer needs to be set.

On the other hand, non-local variable references could be found more quickly.
Normally, if a variable is located two levels higher in the nesting hierarchy, the system must follow two links.
With an extra static link to the grandparent, that variable could be reached in one step.
This improvement reduces the access time for variables in nearby enclosing scopes.

In summary, adding a second static link slightly increases the setup time and memory usage for each activation record, but it speeds up many non-local variable references.
It provides a trade-off between a minor increase in linkage overhead and faster access to variables in shallow nesting levels.