Assignment 8

Qinglin Li, 5110309074

Problem 1

Use the B+-tree to find the first tuple with $branch\_name =$ "Downtown" and $branch\_city <$ "Brooklyn", then scan the following tuples and check whether each tuple satisfies the condition on $assets$. Stop the scanning when $branch\_name \neq$ "Downtown" and $branch\_city \geq$ "Brooklyn"

Problem 2

In a **serial schedule**, no transaction starts until a running transaction has ended.
A **serializable schedule** is a schedule that is equivalent (in its outcome) to a serial schedule

Problem 3

a. If we execute T1 first, the result should be $A = 0$ and $B = 1$
If we execute T2 first, the result should be $A = 1$ and $B = 0$
So in both case, the consistency requirement is satisfied.

b.

T1	T2
Read(A)	
	Read(B)
Read(B)	
	Read(A)
If A=0 then B:=B+1	
	If B=0 then A:=A+1
Write(B)	
	Write(A)

c. No.
Because If we want to produce a serializable result, we must execute the first read instruction after the other's write.

Problem 4

1. To improve throughput and resource utilization.
2. To Reduce waiting time.

Problem 5

a. T31:

lock-S(A)
read(A)
lock-X(B)
read(B)
if A = 0 then B:= B + 1
write(B)
unlock(A)
unlock(B)

T32:

lock-S(B)
read(B)
lock-X(A)
read(A)
if B = 0 then A := A + 1
write(A)
unlock(B)
unlock(A)

b. Yes.
T31 acquire the lock on A first and then T32 acquire the lock on B.

Problem 6

Advantage: It produces only cascadeless schedules, recovery is very easy.
Disadvantage: The set of schedules obtainable is a subset of those obtainable from plain two phase locking, thus concurrency is reduced

Problem 7

Suppose a transaction is rolled back because of a newer transaction's reading or writing the data which it plans to write. If the rolled back transaction is re-introduced with the same timestamp, the transaction should be rolled back again because of the same reason and the process would never end.

Problem 8

A transaction may become the victim of deadlock-prevention rollback arbitrarily many times, thus creating a potential starvation situation.

Problem 9

Volatile storage is storage which fails after a power failure. Caches, main memories are volatile storage.
Non-volatile storage is storage which retains its content after power failures. Magnetic disk, tapes are non-volatile storage.
Stable storage is storage which survives any kind of failure. This type of storage can only be approximated by replicating data.

Volatile memory is the fastest and non-volatile storage is slower. Stable storage is the slowest because of the data replication.

Problem 10

a. Stable storage cannot really be implemented because all storage devices are made of hardware, and all hardware is vulnerable to mechanical or electronic device failures.

b. Database systems approximate stable storage by writing data to multiple storage devices simultaneously. Even if one of the devices crashes, the data will still be available on a different device.

Problem 11

In a system crash, the CPU goes down, and disk may also crash. But stable-storage at the site is assumed to survive system crashes.
In a disaster, everything at a site is destroyed.