**חלק ב'**

סעיף א'

נוכיח:

נניח בשלילה שנוצר deadlock, כלומר קיימות שתי transactions נסמנן: $T_1, T_2$ כך ש-$T_1$ מחכה למנעול כתיבה/קריאה על פריט נתונים x הנעול על ידי $T_2$ ו-$T_2$ מחכה למנעול כתיבה/קריאה על פריט נתונים y הנעול על ידי $T_1$.

אם בה"כ $T_1$ מבקשת מנעול קריאה על פריט נתונים x הנעול לקריאה על ידי $T_2$: מקרה זה לא ייתכן כיוון שלפי מנגנון הנעילות הנתון, אם התנועה $T_1$ מבקשת מנעול קריאה על פריט נתונים x אשר נעול לקריאה על ידי $T_2$ אזי התנועה $T_1$ מקבלת את המנעול ולכן לא יווצר מצב בו היא תחכה לו.

עבור שאר המקרים כאשר בה"כ $T_1$ היא מבקשת המנעול על פריט הנתונים x ($T_1$ מבקשת מנעול כתיבה על פריט נתונים x הנעול לכתיבה/קריאה על ידי $T_2$, או ש-$T_1$ מבקשת מנעול קריאה על פריט נתונים x הנעול לכתיבה על ידי $T_2$)

לפי מנגנון הנעילות, כאשר $ts(T_1) < ts(T_2)$ התנועה $T_2$ מאולצת לשחרר את המנעול והמנעול מועבר ל-$T_1$, כלומר לא יווצר מצב בו $T_1$ מחכה למנעול מ-$T_2$.

לכן מתקיים $ts(T_1) > ts(T_2)$, כלומר לפי המנגנון הנתון התנועה $T_1$ תוכל לקבל את המנעול רק לאחר ש-$T_2$ תשחרר אותו. $T_2$ תשחרר את מנעול הקריאה/הכתיבה שלה על פריט נתונים x כיוון שלוקח זמן סופי לכתוב או לקרוא מ-x. ולאחר ש-$T_2$ תעשה את הפעולה על x היא תשחרר את המנעול כי הוא לא נחוץ לה יותר. כלומר, $T_1$ תקבל את המנעול לאחר זמן סופי ולכן לא ייווצר מצב בו היא תחכה לו במשך זמן אינסופי. סתירה להנחת השלילה שלפיה יש deadlock, לכן מנגנון הנעילות והפרוטוקול 2PLP מונעים קיפאון.

סעיף ב'

נפריך באמצעות דוגמה נגדית:

נגדיר שתי תנועות:

$$T_1: \begin{array}{l} \sim some\ commands... \sim \\ Read(x) \\ Commit \end{array} \qquad T_2: \begin{array}{l} Write(x) \\ if\ (x<0): \\ \quad Abort \\ else \\ \quad Commit \end{array}$$

כאשר $ts(T_1) < ts(T_2)$,

כלומר $T_1$ היא המבוגרת מבין התנועות. נבחר את תחום הערכים של פריט הנתונים $x$ להיות כל המספרים השלמים האי שליליים, ונגדיר את $T_2$ כך שכתבה לפריט הנתונים $x$ את הערך 1- (ערך שלא נמצא בתחום של ערכי $x$) ולכן תבצע Abort. בנוסף נגדיר ש-$T_1$ תגיע לפקודת ה-$Read(x)$ ותבקש מנעול קריאה על פריט הנתונים $x$ בזמן שהוא נעול על ידי $T_2$ ולאחר ש-$T_2$ הספיקה לשנותו (כלומר, לפני שהספיקה לשחררו). לפי מנגנון הנעילות הנתון, $T_2$ (הצעירה יותר) תאלץ לשחרר את המנעול והמנעול יועבר ל-$T_1$, אשר תקרא את ערכו של x לאחר ששונה ע"י $T_2$. כעת, $T_2$ תגיע לשורה האחרונה שלה ותבצע Abort, כלומר השינויים שעשתה בפריט x לא תאמו את הנדרש. לכן נוצרה בעיה: $T_1$ קראה ערך לא נכון של x, ולכן תצטרך לעשות Abort גם כן. מקרה שכזה, שבו Abort של תנועה אחת גורם ל-Abort של תנועות נוספות, נקרא נטישה מדורגת, cascading abort.

(ניתן לחשוב על התנועה $T_2$ כתנועה שכותבת כמה מלאי יש מפריט מסויים בחנות ולכן לא יכול להיות מספר שלילי)