**הגנות:**

* רשימת המשימות – מפני תהליכים בתוך הבריכה
* רשימת המשימות – תהליכים מבחוץ שרוצים להוסיף משימה לבריכה (לא רק התהליך היוצר)
* בריכת החוטים – להגן מפני הריסה כפולה

**פסאודו-קוד:**

תסתכלו בקובץ pseodocode.c בנפרד

**סכנות:**

* גם פונקציית הוספת משימה וגם הפונקציה של החוטים צריכה לקרוא את מצב ה-threadPool (כלומר, האם הוא עומד להיהרס) וגם צריכים לנעול את תור המשימות. יש כאן סכנת LIVELOCK איפשהו, אני די בטוח...  
  לדעתי בעצם לא יהיה LIVELOCK או DEADLOCK אם נממש כמו שצריך את הקריאה המרובה והכתיבה היחידה של המצב state, כי אם פונקציית הוספת המשימה עושה acquire קודם היא משחררת מהר ואם הפונקציה של החוט עושה acquire קודם, היא לא צריכה לחכות לפונקציית ההוספה כדי לקרוא את הערך של state כי אנחנו אמורים לאפשר קריאה בו-זמנית
* אולי יש סיכוי ש:
  + חוט יקרא את read\_state ויגלה שהמבנה במצב ALIVE, ולכן יכנס לתוכן לולאת ה-while
  + CONTEXT SWITCH
  + החוט הראשי מחליט להרוס את המבנה
  + בתוך פונקציית ההריסה, יתבצע broadcast
  + CONTEXT SWITCH
  + החוט (שנכנס לתוך ה-while לפני ה-CS) מגיע לפקודת wait ויחכה לנצח

פתרון אפשרי: לולאה בחוט הראשי: כל עוד יש חוט שלא במצב ZOMBIE, או שיש חוט שלא רוצה כרגע לרוץ (אולי אפשר לבדוק את זה? אולי אפשר לבדוק האם בן שלי נמצא בתור המתנה?) לעשות broadcast. כך, כל החוטים כל הזמן מקבלים שידור שצריך להפסיק לחכות ולהמתין למנעול.