שיעורי בית

1. מדוע בטבלת המצבים- **Thread** איננו חוזר למצב **Running**?
2. האם שימוש ב Thread-ים רבים תמיד משפר ביצועים?
3. מה הרעיון מאחורי ה **ThreadPool**? באילו מקרים נשתמש ב **ThreadPool**? באילו מקרים לא? מדוע לדעתך?
4. מדוע שימוש במשאב כגון **רשימה** יכול ליצור בעיה בתכנות **multi threaded**?
5. אילו **פתרונות** יכולים להיות לבעית פנייה למשאב (כגון רשימה), בעולם ה- multi threaded
6. מה החיסרון בשימוש ב **Monitor.Enter** ו- **Monitor.Exit**? מה היתרון?
7. כיצד **lock** ממומש מאחורי הקלעים (רמז: ראה ILDASM)
8. מהו **critical section**?
9. מהו **deadlock**?
10. מהו **ThreadSafe**?
11. אילו **אוספים** קיימים במחלקת **System.Collection.Concurrent**? מה היתרון בשימוש במחלקות אלה? מה החיסרון?
12. ממש את התוכנית הבאה:
    ייצר מחלקה בשם **ThreadExecutor** ובתוכה שמור **Queue** של **Threads**
    ייצר פונקציות בשם: **Add** אשר מקבלת Thread ומוסיפה אותו לתור (אל תשכח lock)
    ייצר פונקציה בשם **Execute** (אשר גם היא עם lock). פונקציה זאת בודקת אם יש Threads בתור, אם כן- אז עושה להם Start ומחכה לסיומם. היא עושה זאת עבור כל ה Threads ב Queue עד אשר ה- Queue מתרוקן לגמרי. (כלומר כל Thread שביצעה , היא **מסירה** אותו מה- Queue)
    צור ב- **Main** מספר פונקציות, לדוגמא: פונקציה אשר סופרת עד 10, פונקציה המדפיסה "Hello world" וכו', עטוף אותם ב thread והוסף אותם לרשימה
    כעת קרא ל Execute
    *אתגר: בבנאי של **ThreadExecutor** אתחל Thread אשר "מתעורר" פעם בשנייה ובודק אם יש Threads ב- **Queue**, אם כן אז הוא קורא ל **Execute** (ניתן להשתמש ב Timer)
    ** אתגר: פתור שוב את האתגר אך הפעם השתמש ב Thread מתוך ה- **ThreadPool**
    - ממש מחלקה זו שוב אך הפעם השתמש ב- Concurrent Queue (הפעם אין צורך ב lock)