שאלה 1:

ממשים אלגוריתם של מציאת רכיבי קשירות בגרף – ניתן למצוא אלגוריתם יעיל באינטרנט

שאלה 2:

נבצע profiling של האלגוריתם על סוגים שונים של מבנה הנתונים המייצג את הגרף

על מימושים שונים מטריצת שכנויות, רשימה של קשתות, וקטור קודקודים – מימושים שונים נבדוק את היעילות לפי בחירת מבנה הנתונים – ונבחר אותו להמשך השאלות הבאות

שאלה 3:

הוספת פעולות לגרף של הוספה וחיסור של קשתות, והפעלת האלגוריתם למציאת רכיבי קשירות לפני הוספה או הסרה של הקשתות – קלטים יהיו מהלקוחות המכניסים את הנתונים מהקלט והפלט הסטנדרטי.

שאלה 4:

שימוש בצאט beej והקוד של שאלה 3 ליצירת תקשורת של לקוחות עם הקוד.

שאלה 5:

<u>ללא</u> קשר לשאלות הקודמות – Reactor - בניית ספרייה על ידי שימוש בDesign pattern Reactor

שאלה 6:

<u>עם</u> קשר לשאלות הקודמות – לממש את שאלה 4 בעזרת שימוש **Single Thread(async I/O)** בספריה שבנינו בשאלה 5.

שאלה 7:

<u>עם</u> קשר לשאלות הקודמות – לממש את שאלה 4 בעזרת **Multi Threads**

שאלה 8:

<u>עם</u> קשר לשאלות הקודמות – הוספה לספריה של שאלה 5 עם תבנית Reactor להוסיף תבנית Proactor

שאלה 9:

<u>עם</u> קשר לשאלות הקודמות – לממש את שלב 7 בעזרת הספריה שממשנו בשלב 8.

Proactor

Proactor is a thread that listens for incoming connections
(Possibly on multiple sockets)

When a connection occur the proactor starts new thread to handle the connection

REACTOR

- Thread that calls select(2)
- Receives function pointer to call on event
- Allows adding and removing file descriptor
- A thread that calls select(2) and calls the registered function
- Origin POSAv2 and ACE.

Reactor/Proactor/Thread Pool

	Reactor	Thread Pool	Proactor
Number of threads	Always 1	Fixed	No max
advantages	no time wasted on context switch No time wasted on creation and termination overhead or message passing	no creation overhead. Can use multiple processors	Tasks only wait if there are system limitation. not to other tasks. Thread handling time can be very long Time not wasted on message passing
Disadvantages	Can't use multiple processors	message passing overhead	creation overheads
When to use	handling time is short	many tasks are given so creation overhead will be significant. multiple processors can be used	Tasks can hang (for example for user input) and wait for long period of time
When not to use	When task can hang Best performance in multiple	When tasks can hang	When many very short tasks are given (Creation overhead