ii. לחילופין, ניתן להציע את הפתרון הבא המורכב יותר:

```
// x is a shared variable, initially x=1
// each process has a local variable trying
Trying=1
While (trying) {
        While (x < 1)/*empty*/;
        If (fetch-and-add(x,-1) == 1)
                Trying=0;
        Else
                Fetch-and-add(x, 1);
}
Critical section
Fetch-and-add(x, 1);
```

התכונות שפתרון זה מקיים הן (סמנו את כל התכונות הנכונות)

א. הפתרון הזה בעצם דומה לפתרון הקודם מבחינת התכונות שלו

ב. יש מניעה הדדית עבור 2 תהליכים אך לא עבור יותר תהליכים

ג. יש מניעה הדדית עבור כל מספר של תהליכים

ד. אין סכנה של deadlock (או livelock) בתנאי שלמערכת יש רק מעבד אחד

ה. אין סכנה של deadlock (או livelock) עבור כל מספר של מעבדים

ו. יש הגינות במובן הבסיסי של המונח

ז. יש הגינות חזקה: תהליכים נכנסים לקטע הקריטי בסדר שבו הגיעו

ח. הפתרון יעיל בכך שאינו מבוסס על busy waiting

9. להלן הגדרה של סמפור (semaphore):

סמפור (semaphore) הוא_____ (1) _____ לפתירת/לתפעול בעיית הקטע הקריטי (critical section).
בהגדרה זו חסר ביטוי אחד המסומן (1). הביטוי החסר (1) הוא:

א. מערכת חומרה
ב. תכנית מיוחדת עבור מערכת ההפעלה
ג. תכנית מיוחדת הקשורה ל-UPP (אב קדמון)
ד. משתנה אלפא ביתי (טקסט)

10. תהליך הממתין לסמפור בגלל שערך הסמפור שלילי, נמצא במצב:
א. רץ (Running)
ב. מוכן (Ready)
ג. ממתין (Waiting)
ד. סיים (Terminated)

11. בעית החוצץ החסום (bounded buffer) עוסקת בגישת תהליכים למבנה נתונים משותף.

i. סמני את הדרישות לפיתרון הבעיה
א. יצרן וצרכן לא יכולים לגשת לחוצץ בו זמנית
ב. יצרן וצרכן יכולים לגשת לחוצץ בו זמנית בתנאי שמדובר בתאים שונים.
ג. מספר יצרנים לא יכולים לגשת לחוצץ בו זמנית
ד. מספר יצרנים יכולים לגשת לחוצץ בו זמנית בתנאי שמדובר בתאים שונים
ה. אם החוצץ מלא היצרנים נחסמים
ו. אם החוצץ מלא הצרכנים נחסמים
ז. אם החוצץ ריק היצרנים נחסמים
ח. אם החוצץ ריק הצרכנים נחסמים

ii. להלן הצעה לפתרון בעיית החוצץ החסום:

```
//space is a semaphore initialized to n , data is a semaphore initialized to 0
//numbering is 0-based (buf [ 0 ]   to buf [n-1] ), put and get are integers initialized to 0

//producer:
P (space) :
Buf  [  put++  % n  ] = new_item;
V (data) ;

//consumer:
P (data);
My_item = buf [get++  % n  ] ;
V(space) ;
```

סמני את הטענה הנכונה לגבי הפתרון הנ"ל, בהנחה שהסמפורים הוגנים (כלומר תהליך שמבצע P יכנס אחרי מספר סופי של פעולות V);
א. הפתרון נכון אם יש יצרן אחד וצרכן אחד
ב. הפתרון נכון לכל מספר של יצרנים וצרכנים
ג. הפתרון שגוי

12. בעית החוצץ החסום (bounded buffer) עוסקת בשני סוגי תהליכים: האחד מיצר פרטי מידע (product) והשני צורך אותם (consumer) . החוצץ נועד לתאם ביניהם, ובפרט מאפשר ליצרן לאחסן כמה פריטים עד שהצרכן יקח אותם, בשאלה נניח שהחוצץ בגודל 10 פריטים.

פתרון פשוט לבעיה המשתמש בסמפורים הוא:

```
Producer:                 Consumer:
While (1) {               While (1) {
T=new item;               P(full);
P(empty);                 P(mutex);
P(mutex);                 t= Buf[ out++ % 10 ]
Buf[ in++ % 10 ]=t;       V(mutex);
V(mutex);                 V(empty);
V(full);                  <use t>
}                         }
```

א. השימוש בסמפורים הוא כדלהלן (מלאי ערכים או הקיפי בעיגול את התשובה הנכונה)
   i. הסמפור empty מאותחל לערך _____
   ii. הסמפור full מאותחל לערך _____
   iii. הסמפור mutex מאותחל לערך _____
   iv. הפקודה P(empty) נועדה לחסום/לשחרר את התהליך כשהחוצץ ריק/לא ריק/מלא/לא מלא
   v. הפקודה P(full) נועדה לחסום/לשחרר את התהליך כשהחוצץ ריק/לא ריק/מלא/לא מלא
   vi. הפקודה V(empty) נועדה לחסום/לשחרר את התהליך כשהחוצץ ריק/לא ריק/מלא/לא מלא
   vii. הפקודה V(full) נועדה לחסום/לשחרר את התהליך כשהחוצץ ריק/לא ריק/מלא/לא מלא

ב. בהנחה ש- in ו-out משתנים סטטיים משותפים המאותחלים ל-0 , סמני את הטענה הנכונה לגבי המימוש הזה. בשאלה זו נתעלם מהסכנה ל overflow אם מגדילים את in ו-out יותר מדי פעמים:
   i. הפתרון נכון לכל מספר של יצרנים וצרכנים המשתמשים באותו חוצץ במשותף.
   ii. הפתרון שגוי במקרה הכללי, אבל נכון במקרה שיש רק יצרן אחד ורק צרכן אחד
   iii. הפתרון שגוי אפילו במקרה שיש רק יצרן אחד ורק צרכן אחד.

ג. במקרה שבו העתקת הפרטים לוקחת זמן רב, ניתן להשיג יותר מקביליות ע"י הפתרון הבא, שבו ההעתקה עצמה מתבצעת מחוץ לקטע הקריטי:

```
Producer:                 Consumer:
While (1) {               While (1) {
T=new item;               P(full);
P(empty);                 P(mutex);
P(mutex);                 My_out =out++ %10;
My_in =in++ %10;          V(mutex);
V(mutex);                 V(empty);
V(full);                  T=buf [my_out ] ;
Buf  [my_in ] =t;         <use t>
}                         }
```

באותם תנאים כמו השאלה הקודמת, מהי הטענה הנכונה עבור אופטימיזציה זו?
   i. הפתרון נכון לכל מקרה של יצרנים וצרכנים המשתמשים באותו חוצץ משותף.
   ii. הפתרון שגוי במקרה הכללי, אבל נכון במקרה שיש רק יצרן אחד ורק צרכן אחד.
   iii. הפתרון שגוי אפילו במקרה שיש רק יצרן אחד ורק צרכן אחד.

13. המטרה של מנעול מסוג readers-writers היא

א. לתת עדיפות לתהליכים שקוראים את מבנה הנתונים
ב. לתת עדיפות לתהליכים שכותבים (משנים) את מבנה הנתונים
ג. לדאוג להגינות בין קוראים לבין כותבים
ד. להקטין את אילוצי הסנכרון ולאפשר יותר מקביליות במערכת

14. הבעיה של מנעול קוראים/כותבים (readers/writers) עוסקת בגישה של תהליכים למבנה נתונים משותף

i. סמני את כל התכונות הרצויות לפיתרון הבעיה:
א. יכול להיות לכל היותר כותב אחד בקטע הקריטי
ב. יכול להיות לכל היותר קורא אחד בקטע הקריטי
ג. יכולים להיות מספר כותבים בקטע הקריטי בו זמנית
ד. יכולים להיות מספר קוראים בקטע הקריטי בו זמנית
ה. במגבלות של הסעיפים הקודמים לגבי מספר הקוראים והכותבים, הקוראים והכותבים יכולים להמצא בקטע הקריטי בו זמנית.

ii. להלן הצעה לפתרון בעיית המנעול לקוראים/כותבים:

```
//mutex and readers are semaphores initialized to 1

//reder:
P(readers);
P(mutex);
V_all(readers);
  Critical section
V(mutex);

//writer:
P(mutex);
  Critical section
V(mutex);
```

סמני את כל הטענות הנכונות לגבי הפיתרון הנ"ל, בהנחה שהסמפורים הוגנים (כלומר תהליך שמבצע p יכנס אחרי מספר סופי של פעולות V)

א. הפתרון אינו מאפשר ליותר מקורא אחד להכנס לקטע הקריטי בו זמנית
ב. הפתרון אינו מאפשר ליותר מכותב אחד להכנס לקטע הקריטי בו זמנית
ג. הפתרון אינו מאפשר לקורא(ים) וכותב(ים) להכנס לקטע הקריטי בו זמנית
ד. הפתרון עלול לגרום להרעבה של קוראים ו/או כותבים
ה. הפתרון שקול בעצם למנעול רגיל ולכן אינו מנעול קוראים/כותבים

15. לפניך הצעה לפתרון בעית קוראים/כותבים

```
// read and write are semaphores initialized to 1
//reading is initialized to 0
```

```
//Writer
P(read)
P(write)
      Critical Section
V(write)
V(read)
```

```
//reader
If (!reading)  P(read);
Reading++;
     Critical Section
Reading---;
If (!reading) V(read);
```

הבעיה העיקרית בפתרון זה היא (סמני רק אחת)

א. הפתרון אינו מאפשר ליותר מקורא אחד להכנס לקטע הקריטי בו זמנית
ב. הפתרון אינו מאפשר ליותר מכותב אחד להכנס לקטע הקריטי בו זמנית
ג. הפתרון מאפשר לקורא(ים) וכותב(ים) להכנס לקטע הקריטי בו זמנית
ד. הפתרון עלול לגרום להרעבה של קוראים ו/או כותבים
ה. הפתרון שקול בעצם למנעול רגיל ולכן אינו מנעול קוראים/כותבים

16. נתונה משימה המורכבת משלושה תהליכים P0, P1, P2 שרצים במקביל.
נתון כי התהליכים משתמשים בשלושה סמפורים בינאריים S0, S1 , S2 , כמתואר בקטעי הקוד שלהלן:

```
Process P0: {while(true){wait(S0); print '0'; signal (S1); signal(S2);}}
Process P1:{wait(S1); signal (S0);}
Process P2:{wait(S2); signal (S0);}
```

כמו כן, נתון כי הסמפורים מאותחלים באופן הזה: S0=1, S1=0, S2=0.
כמה פעמים ידפיס התהליך P0 את הערך "0"?

א. פעם אחת לפחות
ב. פעמיים בדיוק
ג. פעמיים לפחות
ד. שלש פעמים בדיוק

17. בהמשך לסעיף הקודם , נתון כי הקוד של התהליך P0 נשאר ללא שינוי , ואילו הקוד של התהליכים P1 ו- P2 משתנה באופן שלהלן:

```
Process P1:{ signal (S0); wait(S1); }
Process P2:{ signal (S0); wait(S2); }
```

כמה פעמים ידפיס התהליך P0 את הערך "0"?

א. פעם אחת לפחות
ב. פעמיים בדיוק
ג. פעמיים לפחות
ד. שלש פעמים בדיוק

18. נתונים שני תהליכים P0 ו-P1 ומערך משותף לשניהם bool flag[2]
נתון כי שני איברי המערך מאותחלים ל-false
להלן אלגוריתם למימוש הגישה לקטע הקריטי:

P0	P1

flag[0] = true;
while (flag[1] == true) flag[0] = false;
flag[0] = true;
<קטע קריטי>
flag[0] = false;
	flag[1] = true;
	while (flag[0] == true) flag[1] = false;
	flag[1] = true;
	<קטע קריטי>
	flag[1] = false;

מהו ההיגד הנכון?

א. האלגוריתם מקיים תמיד את הדרישה למניעה הדדית
ב. האלגוריתם לא מקיים תמיד את הדרישה למניעה הדדית
ג. האלגוריתם עלול לגרום להרעבה (starvation)
ד. האלגוריתם עלול לגרום לקיפאון (deadlock)