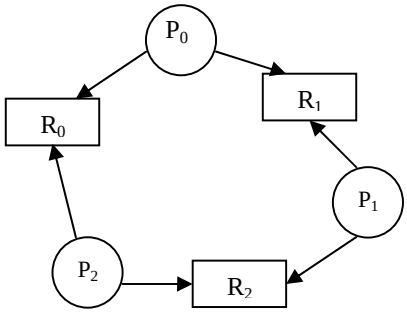
חלק ב' (40 נק')

יש לענות על 2 מתוך 3 שאלות בלבד. אם ענית על כל השאלות ייבדקו רק 2 השאלות הראשונות. יש לכתוב את התשובות במחברת הבחינה בלבד.

1. **(20 נק')** נתונה מערכת הכוללת n תהליכים (processes) $P_0 \ldots P_{n-1}$ ו-n משאבים (resources) $R_0 \ldots R_{n-1}$. תהליכים ומשאבים מאורגנים במעגל כפי שמופיע בציור ל-n=3. לכל תהליך גישה אך ורק לשני המשאבים הסמוכים אליו (כלומר תהליך P_i יכול לגשת למשאבים R_i ו-$R_{i+1 \bmod n}$).



משאבים $R_0 \ldots R_{n-1}$ מיוצגים ע"י מערך בזיכרון משותף:

```
Resource R[n];
```

כאשר טיפוס Resource מוגדר כדלקמן:

```
class Resource {
     void use();
}
```

נגדיר: right(i)=i; left(i)=(i+1)%n.

כל תהליך P_i, $0 \le i < n$, מבצע קוד הבא:

```
while(true) {
    sleep(rand());
    enter();
    work();
    exit();
}
```

כאשר פונקצית work() מוגדרת כדלקמן:

```
void work() {
     R[right(i)].use();
     R[left(i)].use();
}
```

בשאלה הזאת נדון במימוש לפונקציות enter() ו-exit() שיבטיח לתהליך P_i שימוש בלעדי במשאבים R[right(i)] ו-R[left(i)] לכל משך הביצוע של פונקצית work().

<u>סעיף א (9 נק')</u>: לפניך שלושה מימושים לפונקציות `enter()` ו-`exit()` המשתמשים במערך משותף של מנעולים

```
Mutex lock[n];
```

כאשר טיפוס `Mutex` מוגדר כדלהלן:

```
class Mutex {
     void lock();
     boolean try_lock();
     void unlock();
}
```

כאשר פעולות `lock()` ו-`unlock()` הינן זהות ל-P() ו-V() בהתאם ופעולת `try_lock()` הינה זהה ל-P() אך מחזירה false במקרים בהם פעולת P() נתקעת (כלומר במקרים בהם מנעול תפוס).

כל שלושת הפתרונות אינם מספקים תכונת התקדמות (Progress): כלומר הם מאפשרים תרחישים בהם אף תהליך שקורא ל-enter() לעולם לא יצליח לקרוא ל-work(). עליך לציין לכל פתרון האם הוא סובל מ-deadlock או מ-livelock ולנמק את תשובתך ע"י מתן דוגמה מפורטת ל-n=2:

פתרון 1: (3 נק')

```
void enter() {
        lock[right(i)].lock();
        lock[left(i)].lock();
}
```

This solution may result in a deadlock. A scenario for n=2 is as follows:

P0	P1
lock[0].lock()	
	lock[1].lock()
lock[1].lock() gets stuck	
	lock[0].lock() gets stuck

פתרון 2: (3 נק')

```
void enter(){
        while(!lock[right(i)].try_lock());
        while(!lock[left(i)].try_lock());
}
```

This solution is also deadlock prone. As many students justly noted, this solution is exactly the same as the above: It just has the busy wait taken out of lock() to a higher level. Exactly the same scenario as the above results in deadlock – just replace lock() with the loop on try_lock. Note that this is not a livelock. The reason is that once the system gets stuck, there is no possibility for it to get out of this: i.e., any possible schedule will leave the system stuck.

פתרון 3: (3 נק')

```
void enter(){
   while(true) {
     while((!lock[right(i)].try_lock());
     if (!lock[left(i)].try_lock())
        lock[right(i)].unlock();
     else
        return;
   }
}
```

This solution results in a livelock. One of the possible livelock scenarios is as follows:

P0	P1
lock[0].try_lock() returns TRUE	
	lock[1].try_lock() returns TRUE
lock[1].try_lock() fails	
	lock[0].try_lock() fails
lock[0].unlock()	
	lock[1].unlock()
lock[0].try_lock() returns TRUE	
	lock[1].try_lock() returns TRUE
…	…

Note that this is a livelock, because in a real system, the above scenario may persist for infinitely long only theoretically. For example, if P1 is scheduled a moment after P0 unlocks lock[0], it will succeed to take lock[0] and make progress.

פונקצית `exit()` הינה זהה לכל הפתרונות והיא:

```
void exit() {
  lock[right(i)].unlock();
  lock[left(i)].unlock();
}
```

<u>סעיף ב (11 נק')</u>: עליך לספק מימוש שיבטיח תכונת ההתקדמות. על המימוש לתמוך במקבילות מרבית: כלומר תהליכים שאין להם משאבים משותפים יוכלו לבצע `work()` במקביל. אין להשתמש בפרימיטיבים לסנכרון מעבר לאלה המוגדרים בסעיף א' וגם אין להשתמש בפתרונות תכנה לבעיית קטע קריטי (כמו Bakery וכו'). ניתן להניח ש-i ידוע לכל תהליך P_i ו-n ידוע לכל התהליכים.

The idea is to break the symmetry somehow. The easiest way to achieve this is to use the process id (i). For example, we can modify the first solution above in the following way:

```
enter() {
     if (i%2) {
          lock[right(i)].lock();
          lock[left(i)].lock();
     }
     else {
          lock[left(i)].lock();
          lock[right(i)].lock();
     }
}
```

The proof is left as a (good) exercise ☺

2. **(20 נק')** השאלה הזאת עוסקת בתמיכה בזיכרון ווירטואלי (Virtual memory) בחמרה ובתכנה.

א. **(6 נק')** נתונה מערכת זיכרון העובדת בשיטת דפדוף (paging). להלן שלשה סוגי ביטים (bits) הנתמכים ע"י חמרת דפדוף (paging hardware). לכל סוג ביט ציין האם הוא הכרחי לתמיכה בזיכרון ווירטואלי והסבר בקצרה את תפקידו:

i. Present (valid) bit

Present bit indicates whether per virtual memory page indicates whether this page is mapped to a frame in the physical memory. This bit is required because due to the very nature of the virtual memory, at each given moment, only a partial mapping exists between the process virtual address space and the physical memory.

ii. Modified (dirty) bit

Modified bit indicates whether the virtual page contents have been modified within the time interval during which the page was mapped to the physical memory. This bit is not essential. It improves performance by saving the need to write unmodified pages back to disk when they are replaced in the memory. Note also that the performance penalty of not having the modified bit will not be too high as the need to write page back arises only upon its replacement (a rare event for programs obeying the locality principle) and not on each page modification (as some students incorrectly remarked ☺).

iii. Access permission bit

This bit indicates whether the page is Read-only or Read-Write. There was a lot of confusion about this bit. The most common mistake was confusing this bit for the file access permission bits. This bit is essential as it prevents the hardware from writing to the code region pages. The hardware does not have any knowledge whatsoever of read-only pages unless this bit is present. However, since the necessity of this bit allowed for different interpretations (e.g., some students wrote that this bit is always required regardless of the memory management technique used), we decided to cancel the necessity sub-question. So the students who gave a correct (or close to correct) description of this bit functionality, got all the points.

ב. **(6 נק')** תאר כיצד ניתן להשתמש ב- Used bit לכל frame בכדי לממש מנגנון זיכרון ווירטואלי בשיטת דפדוף (paging) ?

Used bit is supplied by hardware to indicate whether a page has been accessed since the last time this bit was set to 0. The hardware can only set this bit to 1. The OS is responsible for zeroing this bit (this depends on how OS actually uses this bit). The bit can be used for approximating the LRU page replacement. The Clock page replacement algorithm is the most notable example.

ג. **(8 נק')** עליך להשתמש באלגוריתם לתמיכה בזיכרון ווירטואלי בשיטת דפדוף (paging) שדורש את ה-Used bit. אולם החומרה עליה רצה מערכת ההפעלה לא מספקת את הביט הזה. הסבר איך ניתן לסמלץ את ה-Used bit בתוכנה בצורה היעילה ביותר האפשרית.
רמז: העזר באחד הביטים הכרחיים מסעיף א' והתמיכה הניתנת ע"י החמרה בביט זה.

The Used bit can be simulated using the Valid (Present) bit which is always supported by the paging hardware (see above). This is done as follows:

1. Since the validity information is lost (we are overriding the valid bit), the OS must maintain an internal record of the valid pages.
2. Whenever a page is brought to memory, the OS zeroes the valid bit.
3. Whenever a page replacement algorithm based on the Used bit dictates to assign zero to it, the OS zeroes the valid bit.
4. Whenever hardware accesses a page whose valid bit equals 0, a page fault interrupt is generated as usual, and the OS handler is called.
5. The OS page fault handler does the following:
    i. Check with its internal records whether this page is mapped or not.
    ii. If the page is not mapped, schedule disk I/O and block the process as usual.
    iii. If the page is mapped, set its valid bit to 1.

Note that the above method gives a precise simulation of the Used bit without losing the valid bit functionality. The modified bit cannot be used for this purpose as it does not give an indication whether a page was accessed if the access did not result in a modification. E.g., the pages in the code region are never modified. On the other hand, they are most frequently accessed, and therefore, must be kept in the physical memory most of the time for good performance.

The students, who described the above method (almost) precisely, got all the points. The students who did not provide all the details (e.g., did not indicate that the need of the internal record of the valid pages), got some points (usually at most 2) deducted. The students who indicated that the valid bit should be used and threw some raw ideas, got some points (usually, 2-3). The students who suggested to use Modified/Access Permission bit, or were too far from the correct solution did not get any points.