חלק א (55 נקודות)

ענו על **שלוש** השאלות הבאות.

שאלה 1 (20 נקודות)

פרוטוקול סינכרון bsync–(barrier synchronization) המוגדר כדלקמן :

P = {P1, …,Pn} – קבוצת תהליכים בגודל n.

נסמן ב-bsync(i,k) את ההפעלה ה k-ית של שלהפרימיטיב bsync ע"י התהליך Pi (k> 0). פעולת bsync(i,k) איננה חוזרת עד שכל תהליך Pj, 1<=j<=n, יקרא k פעמים ל bsync.

במילים אחרות, תהליך נתון "מסתכל" כמה פעמים התבצע bsync ע"י תהליכים אחרים ביחס לכמות הפעמים שהוא הפעיל את הפעולה. כל הפעלה של bsync מבטיחה שתהליך לא ממשיך עד ששאר התהליכים ביצעו כמות זהה של קריאות ל bsync.

למשל, בדוגמא הבאה שבה n=3, פעולה bsync בשורה 6 בקוד של תהליך P2 לא תחזור עד שתהליך P1 יקרא לפעולה bsync בשורה 8 ותהליך P3 יקרא לפעולה bsync בשורה 7.

line	P3	P2	P1
1	…	bsync	…
2	…	…	…
3	bsync	…	…
4	…	…	…
5	…	…	…
6	…	bsync	…
7	bsync	…	Bsync
8	…	bsync	Bsync
9	bsync	…	Bsync

(המשך השאלה בעמוד הבא)

(10 נק') א. כתבו פסואדו-קוד ל-bsync למקרה שבו כל תהליך מפעיל את bsync פעם אחת בלבד. השתמשבו ב test-and-set-locked אטומית ופעולות read/write אטומיים בלבד.

(10 נק') ב. כתבו פסואדו-קוד ל-bsync למקרה שבו כל תהליך מפעיל את bsync מספר כלשהו של פעמים. השתמשבו ב סמפורים ופעולות read/write אטומיים בלבד.

שאלה 2 (35 נקודות)

(10 נק') א. עליכם לממש <u>רשימה מקושרת חד כיוונית ציקלית (מעגלית)</u>. הרשימה תהיה בעלת איבר סרק (dummy node) יחיד, אשר עליו מצביעים מראש הרשימה. בצורה זאת אין צורך לבדוק את מקרה הקצה של הוצאת איבר אחרון מהרשימה או הוספת איבר ראשון – ברשימה תמיד יש איבר אחד לפחות.

השלימו את הקוד הבא להכנסת איבר לראש הרשימה. עליכם להשלים גם את מבני הנתונים המגדירים את הרשימה ואיבר הרשימה. ניתן להשתמש במנעול אחד בלבד. קוד לא יעיל (מבחינת מספר המשתנים או הביצועים) יקבל ניקוד חלקי בלבד.

```
typedef struct Node_t {
        void* data;
        Node *next;

} Node;


typedef struct List_t {
        Node head; // dummy empty node. Always first
        _____________________
        _____________________
        _____________________

} List;
```

```
void insert(List list, void* data){
        Node* node = (Node*)malloc(sizeof(Node));
        node->data=data;
```

(15 נק') ב. נסתכל על מימוש של <u>פונקציית remove מרשימה מקושרת דו כיוונית ציקלית (מעגלית)</u>. הרשימה נועדה לשפר את הגישה המקבילית של ממספר תהליכונים (threads) לרשימה, ולכן משתמשת במנעולים ברמה של איברי הרשימה.

```
typedef struct Node_t {
        void* data;
        Node *next, *prev;
        Lock lock;
} Node;

typedef struct List_t {
        Node head, tail; // dummy empty nodes
} List;

InitList(){
        head->prev=head->next=tail;
        tail->prev=tail->next= head;
}

void  remove(List* list, Node* node)
{
```

```
Node* prev;
if (node==NULL || node->prev==NULL || node-next==NULL)
    return;
lock(node->prev->lock);
lock(node->lock);
lock(node->next->lock);
        node->prev->next = node ->next;
        node->next ->prev = node ->prev;
        prev=node->prev;
        node->next=NULL;
        node->prev=NULL;
        unlock(prev ->next->lock);
        unlock(node->lock);
        unlock(prev->lock);
}
```

במימוש זה קיימת בעיה. מצאו את הבעיה והסבירו מדוע היא יכולה להיגרם ובאילו תנאים.

(10 נק') ג. כעת נחליף את פונקצית ה-remove מסעיף ב בפונקציה הבא:

```
void  remove(List* list, Node* node)
{
        Node* prev;
        if (node==NULL) return;
        lock(node->lock);
        if (node==NULL || node->prev==NULL || node-
>next==NULL){
                unlock(node->lock);
                return;
        }
        lock(node->prev->lock);
        lock(node->next->lock);
        node->prev->next = node ->next;
        node->next ->prev = node ->prev;
        prev=node->prev;
        node->next=NULL;
        node->prev=NULL;
        unlock(prev ->next->lock);
        unlock(node->lock);
        unlock(prev->lock);
}
```

הוכיחו או הפריכו את ייתכנות ה deadlock בעקבות השימוש בפונקציה זו.

חלק ב (25 נקודות)

ענו על **חמש** השאלות הבאות. משקל כל שאלה 5 נקודות.

שאלה 4

הסבירו כיצד ממומשות קריאות מערכת (system calls) באמצעות int 0x80. אפשר להביא דוגמא של מימוש קריאת מערכת כלשהי כפי מתואת במדריך הלמידה.

שאלה 5

מהו i-node? פרטו את השדות שמכיל מבנה ה i-node.

שאלה 6

הסבירו מהו RAID2 ומה יתרונו על שיטת RAID1.

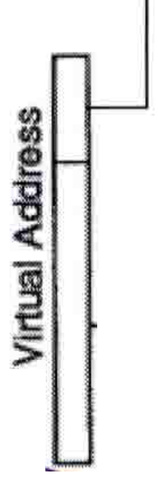
שאלה 7

נתון מנגנון תזמון תהליכים בשיטת התגרלה (lottery scheduling). כיצד, בלי לשנות קוד של מתזמן, ניתן לקבל מהמנגנון הזה תזמון תהליכים שייתן לתהליכים המתוזמנים זמני תגובה כמו round robin?

שאלה 8

השלימו את השרטט. כיצד מתבצע תרגום כתובת לוגית לכתובת פיזית באמצעות page table?

שרטוט:



חלק ג (20 נקודות)

ענו על **ארבע** שאלות רב-ברירה (אמריקאיות). משקל כל שאלה 5 נקודות.
בכל שאלה יש לבחור את התשובה הנכונה ולהקיף בעיגול את אות התשובה שבחרתם.

שאלה 9

מערכת הקבצים של מערכת הפעלה מסוימת משתמשת בשיטת ה-I-node.

- גודל הבלוק במערכת הקבצים הוא 1 Kbyte
- כתובת הבלוק בדיסק היא 4 בתים (bytes)
- 12 שדות של ה-I-node יכולים להחזיק ישירות כתובת הבלוק בדיסק
- שדה נוסף אחד נועד להחזיק כתובת של ה-single indirect block
- עוד שדה נוסף אחד נועד להחזיק כתובת של ה-double indirect block
- ועוד שדה נוסף אחד נועד להחזיק כתובת של ה-triple indirect block

גודלו של קובץ מסוים במערכת 1000 Kbyte. מהי כמות הבלוקים שדרושה להחזקת קובץ זה

במערכת הקבצים (לא כולל את הבלוק שמכיל את ה- i-node של הקובץ)?

- 1000
- 1005
- 1010
- 1011

המשך הבחינה בעמוד הבא

שאלה 10

נתונים שני תהליכים שרצים במקביל. להלן הפסאודו-קוד של להם:

Process 1	Process 0

while (1){
for (i=0; i<N; i++)
down(Si);
/* Critical section */
for (i=N-1; i>=0; i--)
up(Si);
}
	while (1){
	for (i=0; i<N; i++)
	down(Si);
	/* Critical section */
	for (i=N-1; i>=0; i--)
	up(Si);
	}

כאשר Si –ים הם N סמפורים בינאריים שאותחלו ל 1. N הוא מספר טבעי גדול מ 2.

בחר את הטענה הנכונה:

- שני תהליכים יכולים לשהות בו זמנית בקטע קריטי.
- שני תהליכים עלולים להיכנס למצב קיפאון.
- הפרוטוקול מבטיח קדימות של Process 0 על פני התהליך המתחרה.
- הפרוטוקול מבטיח קדימות של Process 1 על פני התהליך המתחרה.
- הפרוטוקול פוטר את בעיית הקטע הקריטי.

שאלה 11

בחרו סיגנל (signal) אשר אי-אפשר להתעלם ממנו (באמצעות SIG_IGN):

- SIGINT
- SIGKILL
- SIGSEGV
- SIGALRM

שאלה 12

מהו החיסרון המובהק של שיטת ה-rendezvous לעומת ה-mailbox ב-message passing?

- שיטת ה-rendezvous קשה יותר למימוש במערכת הפעלה.
- שיטת ה-rendezvous פחות גמישה מכיוון שהתהליכים (השולח והנמען) הופכים להיות מסונכרנים (תלויים זה בזה).
- בשיטת ה-rendezvous לא ניתן לממש מניעה הדדית בגישה לקטעים קריטיים (critical sections).
- שיטת ה-rendezvous גורמת לסחרור (thrashing)

ב ה צ ל ח ה !