



מבוא למדעי מחשב מ' / ח' (234117 / 234114)

סמטר אביב תשס"ז

מבחן מסכם מועד א', 12 ספטמבר 2007

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
שם פרטי	שם משפחה	מספר סטודנט							

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה.
- בדקו שיש 27 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתוב תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר.
- אין לכתוב הערות והסברים לתשובות אם לא נתבקשתם מפורשות לכך.
- בכל השאלות, הינכם רשאים להגדיר (ולממש) פונקציות עזר כרצונכם.
- אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צוין אחרת בשאלה.
- פתרון שלא עומד בדרישות הסיבוכיות יקבל ניקוד חלקי בלבד.

צוות הקורסים 234114/7
מרצים: סאהר אסמיר, ד"ר אלי בן ששון (מרצה אחראי).
מתרגלים: מירי בן-חן, אלעד הרמתי, עידו חניאל, אסנת טל, ששה סקולוזוב, אנדרי קלינגר, איתי שרון (מתרגל אחראי).

שאלה	ערך	הישג	בודק
1	25		
2	25		
3	25		
4	25		
סה"כ	100		

בהצלחה!

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.



שאלה 1 (25 נקודות)

סעיף א

נתונה תוכנית ה-C הבאה:

```
#define N 5

void mish(char** arr1, char** arr2, int n)
{
    char *p;
    if((n<=0) || (arr1 == arr2)) return;
    p=*arr1;
    *arr1=*arr2;
    *arr2 = p;
    mish(arr1+1, arr2-1, n-1);
}

void mash(char* arr[][N], int n)
{
    if(n>N/2) return;

    mish(&arr[n][0], &arr[N-n-1][N-1], N);
    mash(arr, n+1);
}

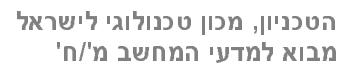
int main()
{
    int i, j;
    char* arr[][N] = {{"It", "is", "the", "end", "of"},
                      {"the", "world", "as", "we"}, {"know", "it", "and"},
                      {"I", "feel"}, {"fine"}};

    mash(arr, 0);
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            printf("%s%c", arr[i][j]? arr[i][j] : "",
                    (j==N-1)? '\n' : '\t');

    return 0;
}
```

1. מה הם הפרמטרים אותם מקבלת הפונקציה `mish()` ומה מבצעת הפונקציה?

2. מה תדפיס התכנית?



- 4 -



סעיף ב

בכל אחד מהסעיפים הבאים מופיעות מספר שורות קוד. לכל קטע קוד, הקיפו בעיגול את התיאור המתאים:

- א. **ללא שגיאות** – הקוד יתקמפל ללא כל שגיאה וירון ללא תקלות.
ב. **שגיאת זמן ריצה** – הקוד יתקמפל ללא שגיאות, אולם הוא עלול לבצע שגיאה בזמן הריצה.
ג. **שגיאת קומפילציה** – הקוד לא יעבור קומפילציה.

1.

```
int arr[]={1,2,3,4,0};  
while(*arr)  
    printf("%d ", *(arr++));
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

2.

```
int i[8] = {0,1,2,3,4,5}, *p = i+4;  
*(p + 4) = 8;
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

3.

```
int i=1, *j=&i+1;  
(double(i/2)) ? *j=2 : *(j-1)=2;
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

4.

```
void f(int x) {  
    --x;  
}  
void h(int n) {  
    if(!n) return;  
    h(f(n));  
}  
int main() {  
    h(3);  
    return(0);  
}
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

5.

```
char s[5]={'H','e','l','l','o'}, *p=s;  
do {  
    putchar(++(*p));  
} while (*(++p));
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

[illegible]

[illegible]

- 9 -

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.



שאלה 3 (25 נקודות)

בעיית מספרי הנוי דומה מאוד לבעיית מגדלי הנוי: ישנם 3 מוטות ו- n טבעות המסודרות בהתחלה על אחד המוטות, מהגדולה (בתחתית) לקטנה (בפסגה). ההבדלים:

- **הטבעות ממוספרות:** מספר הטבעת הגדולה ביותר n ומספר הטבעת הקטנה ביותר 1
- **המגדלים ממוספרים:** לכל אחד משלושת המגדלים יש מספר (0, 1 או 2)
- **המצב הסופי הדרוש** הוא שכל טבעת תהיה ממוקמת על המוט שמספרו הוא כשארית חלוקת מספר הטבעת ב-3. כלומר הטבעות 3,6,9 ... על מוט מספר 0, טבעות 1,4,7 ... על מוט מספר 1 וטבעות 2,5,8 ... על מוט מספר 2.

חוקי המשחק נשמרים:

1. מותר להזיז רק טבעת בודדת בכל צעד
2. אסור למקם טבעת מעל לטבעת קטנה ממנה

לנוחותכם, מובא להלן הקוד של בעיית הנוי המקורית, כפי שנלמד בכיתה:

```
typedef enum{A,B,C} tower_t;

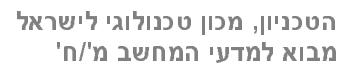
void hanoi(int n, tower_t from, tower_t to)
{
    tower_t via = (A + B + C) - from - to;
    if (n == 0) return;
    hanoi(n-1, from, via);
    printf("Move disc %d from %c to %c\n",
           n, 'A' + from, 'A' + to);
    hanoi(n-1, via, to);
}
```

סעיף א'

כתבו פונקציה **רקורסיבית** הפותרת את בעיית מספרי האנוי עם **הקלה**: חוק מספר 1 לעיל מתבטל, כלומר **ניתן** להזיז מספר טבעות עליונות בבת אחת (תוך שמירה על הסדר היחסי שלהן) ממגדל למגדל. הזזה של מספר טבעות ניתן לעשות אך ורק ע"י קריאה לפונקציה `move_several` אשר חתימתה:

```
void move_several(int n, tower_t from, tower_t to)
```

על הפונקציה שלכם לגרום להדפסת סדרת ההזזות הדרושה. שימו לב שהפונקציה `move_several` מדפיסה הודעה המתאימה להזזות שהיא מבצעת, ולכן רק במקרה בו אתם מזיזים טבעת שלא ע"י קריאה ל-`move_several` עליכם להדפיס הודעה מתאימה.

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.



```
void hanoi2(int n, int loc)
```

```
{
```

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.

כתבו פונקציה רקורסיבית הפותרת את בעיית מספרי הנוי תוך קיום כל הדרישות, בפרט דרישה 1 לעיל.

רמז: חשבו במה ניתן להחליף את הקריאה לפונקציה `move several`.

{

This image shows a single page of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page, leaving small margins at the top and bottom. There is no handwriting or other markings on the paper.



שאלה 4 (25 נקודות)

נוסחת CNF נקראת **דלילה** אם כל משתנה מופיע בעשר פסוקיות שונות לכל היותר. מטרת שאלה זו היא לכתוב גרסא יעילה יותר של פונקציית DPLL המיועדת לפתור את בעיית הספיקות (אך ורק) על נוסחאות דלילות. לאורך השאלה, מספר המשתנים יסומן ב-N ומספר הפסוקיות ב-M (שני המספרים הם מספרים שלמים ומוגדרים ע"י #define). הגדרתה של נוסחת CNF מופיעה בתחתית העמוד, כתזכורת.

לצורך שמירת נוסחת ה-CNF והצבות חלקיות לנוסחה נשתמש בטיפוס truth שהוגדר בכיתה כך:

```
typedef enum { FALSE, UNSET, TRUE } truth;
```

נוסחת ה-CNF נשמרת במערך דו-מימדי CNF[m][N] truth וכל שורה בו מייצגת פסוקית כך:

- $CNF[i][j] = \text{TRUE}$ אם הליטרל x_j שייך לאילוף ה-i
- $CNF[i][j] = \text{FALSE}$ אם הליטרל x_j שייך לאילוף ה-i
- $CNF[i][j] = \text{UNSET}$ כאשר גם x_j וגם $\neg x_j$ אינם מופיעים כלל באילוף ה-i

הצבה חלקית נשמרת במערך truth assignment[N].

הקוד של אלגוריתם DPLL שהוצג בכיתה מובא לנוחיותכם.

תזכורת: נוסחת CNF היא מהצורה $C_0 \text{ AND } C_1 \text{ AND } \dots \text{ AND } C_{M-1}$ כאשר:

- כל C_i נקראת **פסוקית**, והינה מהצורה: $C_i \equiv (L_j \text{ OR } L_k \text{ OR } \dots \text{ OR } L_l)$
- כל L_j נקרא **ליטרל**, והינו משתנה בולאני (x_j) או ההופכי שלו ($\neg x_j$).

דוגמא לנוסחת CNF:

$$(x_1 \text{ OR } x_4) \text{ AND } (\neg x_2 \text{ OR } x_0 \text{ OR } \neg x_3) \text{ AND } (\neg x_4) \text{ AND } (\neg x_1 \text{ OR } x_3 \text{ OR } x_0 \text{ OR } \neg x_4)$$

הצבה היא קביעת ערך (TRUE או FALSE) למשתנים הבוליאניים. הצבה חלקית משמעותה קביעת ערך לחלק מן המשתנים. הצבה חלקית **מספקת** את הנוסחה אם ורק אם בכל פסוקית, יש לפחות ליטרל אחד אשר מקבל ערך TRUE.

לדוגמא: ההצבה החלקית $x_0=x_1=\text{TRUE}$, $x_2=x_3=\text{UNSET}$, $x_4=\text{FALSE}$ מספקת את הנוסחה שלעיל, ולעומת זאת ההצבה החלקית $x_0=x_1=x_2=x_3=\text{UNSET}$, $x_4=\text{TRUE}$ אינה מספקת הנוסחה, שכן הפסוקית השלישית מקבלת ערך FALSE תחת הצבה זו.



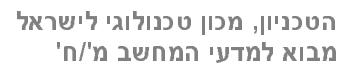
- 18 -



```
truth DPLL(truth CNF[][N], truth assignment[], int m, int i)
// i denotes the smallest unassigned variable. To use the
// function call DPLL(CNF, assignment, m, 0).
{
    truth current_truth;
    current_truth = CNF_sat(CNF, assignment, m);
    if (current_truth==TRUE)
        return TRUE;
    if (current_truth==FALSE)
        return FALSE;
    assignment[i]=FALSE;
    if (DPLL(CNF, assignment, m, i+1)==TRUE)
        return TRUE;
    assignment[i]=TRUE;
    if (DPLL(CNF, assignment, m, i+1)==TRUE)
        return TRUE;
    assignment[i]=UNSET;
    return FALSE;
}

truth CNF_sat (truth CNF[][N], truth assignment[], int m)
{
    truth t=TRUE;
    int i;
    truth clause_truth;
    for (i=0; i<m; i++){
        clause_truth=clause_sat(CNF[i], assignment, N);
        if (clause_truth==FALSE)
            return FALSE;
        if (clause_truth==UNSET)
            t=UNSET;
    }
    return t;
}

truth clause_sat (truth clause[], truth assignment[], int n)
{
    int j;
    truth t=FALSE;
    for (j=0; j<n; j++) {
        if (clause[j]!=0) {
            if (assignment[j]==clause[j])
                return TRUE;
            if (assignment[j]==UNSET)
                t=UNSET;
        }
    }
    return t;
}
```



- 20 -

סעיף א'

לצורך שיפור זמן הריצה על נוסחאות דלילות, נעשה שימוש במערך `int appears[N]` [10] אשר יציין לכל משתנה את מספריהן של הפסוקיות בהן הוא מופיע. כתבו קוד בעל סיבוכיות זמן ריצה קטנה ככל האפשר לפונקציה הבאה המאתחלת את המערך `appears`:

```
truth appears init(truth CNF[][N], int m, int appears[][10])
```

לאחר היציאה מהפונקציה על השורה ה-i של המערך appears להכיל את מספריהן של עשר הפסוקיות בהן מופיע המשתנה ה-i. אם משתנה זה מופיע בפחות מעשר פסוקיות, על התאים הנותרים להכיל את הערך 1-. על הפונקציה להחזיר ערך TRUE אם הנוסחה היא דלילה. אחרת, עליה להחזיר ערך FALSE.

```
truth appears_init(truth CNF[][N], int m, int appears[][10])
{
```

This image shows a full page of blank, lined paper. It features approximately 20 evenly spaced horizontal grey lines across its entire width, providing a template for handwriting practice or general note-taking. The margins are consistent on all sides.



הטכניון, מכון טכנולוגי לישראל מבוא למדעי המחשב מ'ח'

[illegible]



סעיף ב'

עדכנו את הקוד של פונקציית `CNF_sat` שהוצגה בכיתה כך שהפונקציה תבדוק רק את הספיקות של הפסוקיות המכילות את המשתנה האחרון שהוצב. סיבוכיות זמן הריצה צריכה להיות $O(N)$. הפונקציה תחזיר ערך `FALSE` אם ורק אם אחת הפסוקיות הללו מקבלת ערך `FALSE`. בכל מקרה אחר הפונקציה תחזיר ערך `UNSET`. הפונקציה תקבל כקלטים נוספים את מספר המשתנה האחרון לו הוצב ערך (`last`) ואת המערך `appears` מסעיף א'. ניתן להניח כי המערך מאותחל כמתואר בסעיף א', גם אם לא פתרתם את סעיף א'. מותר לעשות שימוש בקוד `DPLL` המוצג לעיל.

```
truth CNF_sat2(truth CNF[][N], truth assignment[], int m,  
               int appears[][10], int last)  
{
```



- 24 -

[illegible]

[illegible]

This image shows a single page of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.