את"ם – תרגיל בית מס' 3
סמסטר אביב תשע"ח

תאריך פרסום: 20/5/2018 תאריך הגשה: 7/6/2018 (בשעה 23:59)

מתרגל אחראי על התרגיל: יותם זיו (yotam.ziv6@gmail.com)

- **ההגשה בזוגות בלבד לתא ההגשה של הקורס ובאמצעות הגשה אלקטרונית.**
- **שאלות על התרגיל יש להפנות ליותם זיו במייל עם מספר הקורס בנושא ההודעה.**
- **הגשות באיחור יש לתאם עם המתרגל האחראי של הקורס (לא של התרגיל) לפני מועד ההגשה הכללי.**
- **אין להגיש לתא הקורס לאחר מועד ההגשה.**
- **עדכון אחרון: 31.5.2018 – מסומן בצהוב**

נושא התרגיל: שגרות + רקורסיה + מבנים.

בתרגיל זה שני חלקים:

- חלק א' מכיל שתי שאלות, עליהן עליכם לענות בכתב ולהגיש לתא הקורס (יש להדפיס את טופס התרגיל ולענות על גביו). אין צורך להדפיס את שאר התרגיל, רק את החלק היבש.
- חלק ב' דורש כתיבת קוד בשפת האסמבלי של PDP-11, כפי שנלמדה בהרצאות ובתרגולים. את הקוד יש לכתוב בקובץ ex2.s11, עם תיעוד פנימי ולהגיש אלקטרונית. הוראות הגשה מפורטות נמצאות **בסוף התרגיל**.

חלק א' – יבש

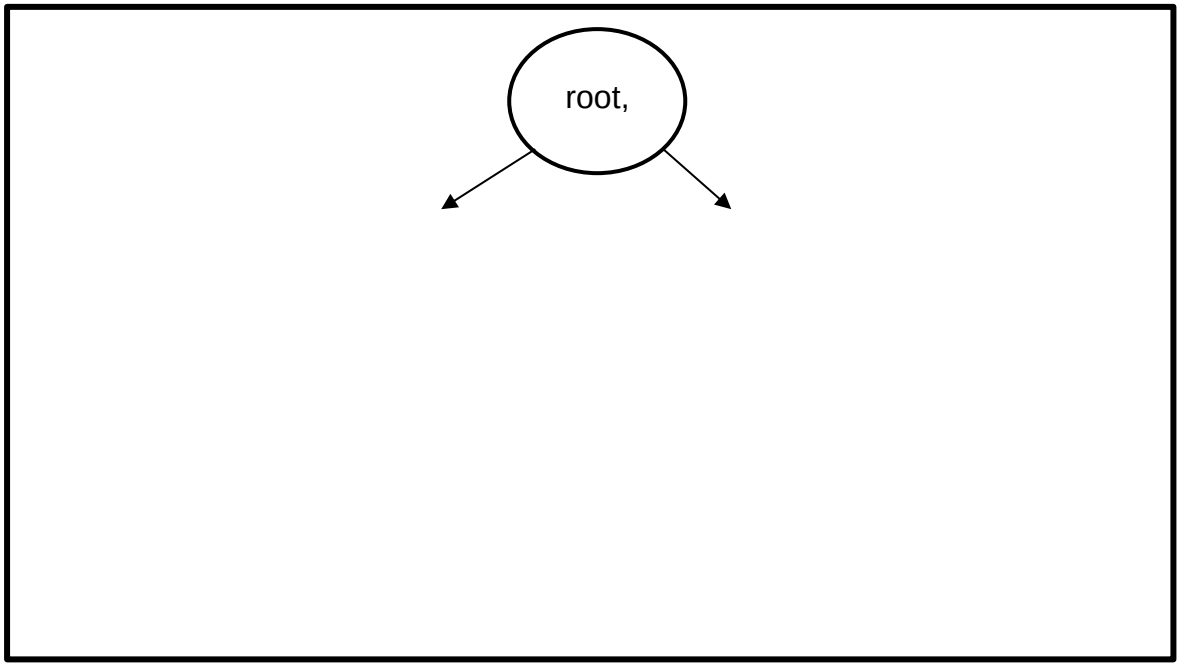
לפניכם קטע התוכנית הבא. ענו על הסעיפים שבעמודים הבאים.

```
.01          .       =       torg+1000
.11          NIL     =       0
12.main:
13.          mov     #main,  sp
14.          mov     #root,  –(sp)
15.          jsr     pc,     bar
16.          tst     (sp)+
17.          halt

20.bar:
21.          clr     c
22.          clr     r0
23.          clr     r1
24.          mov     2(sp),  –(sp)
25.          jsr     pc,     foo
26.          mov     (sp)+,  r1
27.          sxt     r0
28.          tst     c
29.          beq     end_bar
30.          div     c,      r0
31.end_bar:  rts     pc
32.c:        .blkw   1
```

```
60.  foo:
61.          tst     2(sp)
62.          beq     end_foo
63.          mov     @2(sp),  –(sp)
64.          jsr     pc,      buz
65.          inc     c
66.          add     #2,      4(sp)
67.          mov     @4(sp),  –(sp)
68.          jsr     pc,      foo
69.          add     (sp)+,   (sp)
70.          add     #2,      4(sp)
71.          mov     @4(sp),  –(sp)
72.          jsr     pc,      foo
73.          mov     (sp)+,   4(sp)
74.          add     (sp)+,   2(sp)
75.  end_foo: rts    pc

80.          .       =        torg+5000
81.  root:   .word   3,  le1,  ri1
82.  le1:    .word   17, NIL,  NIL
83.  ri1:    .word   10, le2,  NIL
84.  le2:    .word   6,  NIL,  NIL
```

```
40.buz:
41.            mov   2(sp),    –(sp)
42.            clr   4(sp)
43.loop_buz:   tst   (sp)
44.            beq   end_buz
45.            asl   (sp)
46.            bcc   loop_buz
47.            inc   4(sp)
48.            br    loop_buz
49.end_buz:    tst   (sp)+
50.            rts   pc
```

1. ציירו את העץ המוגדר בין שורות 84–81 **לפני ריצת התוכנית**. לכל צומת בעץ, ציינו את התווית המייצגת את הצומת וכן את הערך של הצומת (על-פי הדוגמה).



2. עבור כל אחד מהפרמטרים אותם מקבלת השיגרה **bar**, רישמו בטבלה הבאה מהו שטח ההעברה שלו, ציינו אם הוא משמש לקלט, לפלט, או גם לפלט וגם לקלט, וכן האם הוא מועבר לפי ערך או לפי כתובת. אם הפרמטר מועבר דרך אוגר, ציינו מיהו האוגר. שימו לב כי ייתכן שתישארנה בטבלה שורות ריקות.

פרמטר מספר	שטח העברה	קלט / פלט / שניהם	ערך / כתובת
1			
2			
3			
4			

3. מה מבצעת השיגרה **buz** ? סמנו תשובה אחת נכונה.

א. **מחסרת** ממילת פרמטר הקלט את מספר הסיביות הכבויות בה ומחזירה ערך זה.
ב. **מחסרת** ממילת פרמטר הקלט את מספר הסיביות הדולקות בה ומחזירה ערך זה.
ג. **מחברת** למילת פרמטר הקלט את מספר הסיביות הכבויות בה ומחזירה ערך זה.
ד. **מחברת** למילת פרמטר הקלט את מספר הסיביות הדולקות בה ומחזירה ערך זה.
ה. **מחזירה** את מספר הסיביות הכבויות במילת פרמטר הקלט.
ו. **מחזירה** את מספר הסיביות הדולקות במילת פרמטר הקלט.

4. מה יהיה תוכן המחסנית **מיד לאחר** הקריאה לשיגרה **foo** בשורה 68, אם ידוע כי הפרמטר שהועבר לשיגרה דרך המחסנית היה הכתובת של התווית le2? ניתן לכתוב ביטוי מהצורה "הכתובת של שורה 15" וניתן להשתמש בתוויות. הניחו כי בתחילת התוכנית תוכן של אוגר r_i הוא i (עבור $i<6$). אם קיים במחסנית ערך לא ידוע, כיתבו "לא ידוע" במקום המתאים.

תוכן (מספר אוקטאלי)	כתובת
	754
	756
	760
	762
	764
	766
	770
	772
	774
	776
706201	1000

5. מה יהיה תוכן האוגרים r0 ו-r1 בסיום ריצת התוכנית? כיתבו את הערכים בבסיס אוקטאלי:

אוגר	ערך
r0	
r1	

6. נניח (לצורך סעיף זה בלבד) כי היו משנים את שורה 83 ל:

ri1: .word 10, le2, root

סמנו בעיגול משפט אחד נכון:

א. לא היה חל שינוי בריצת התוכנית.
ב. התוכנית הייתה נכנסת ללולאה אינסופית.
ג. הייתה קורית שגיאה בזמן ריצה.
ד. הייתה קורית שגיאה בזמן תרגום.
ה. הערכים הסופיים באוגרים r0 ו-r1 היו משתנים.
ו. אף אחת מהתשובות א – ה אינה נכונה.

נמקו את תשובתכם בקצרה. בפרט, אם סימנתם את משפט ג או ד, ציינו את סיבת השגיאה, ואם סימנתם את משפט ה, רישמו (בבסיס אוקטאלי) את הערכים הסופיים שהיו מקבלים האוגרים r0 ו-r1.

_____

_____

7. נניח (לצורך סעיף זה בלבד) כי היו משנים את שורה 14 ל:

mov #NIL, –(sp)

סמנו בעיגול משפט אחד נכון:

א. לא היה חל שינוי בריצת התוכנית.
ב. התוכנית הייתה נכנסת ללולאה אינסופית.
ג. הייתה קורית שגיאה בזמן ריצה.
ד. הייתה קורית שגיאה בזמן תרגום.
ה. הערכים הסופיים באוגרים r0 ו-r1 היו משתנים.
ו. אף אחת מהתשובות א – ה אינה נכונה.

נמקו את תשובתכם בקצרה (על-פי ההנחיות של סעיף 6).

_____

_____

8. רישמו (בבסיס אוקטאלי) במה יש להחליף את הערך 17_8 שבמילה בכתובת le1 (שורה 82) כך שבסיום התוכנית, הערך של r0 יהיה 5 והערך של r1 יהיה 1:

le1: .word [ ] **, NIL, NIL**

חלק ב' – רטוב (דמקה)

תיאור המשימה

בתרגיל זה תתבקשו להרחיב את התוכנית שכתבתם בתרגיל בית 2 ולהמשיך במימוש **המערכת שמנהלת משחק דמקה** . בסיום חלק זה, התוכנית שלכם לא רק תמצא את כמות המהלכים האפשריים לפי מצב נתון אלא תחשב מה הוא המהלך האופטימאלי לביצוע בשלב זה של המשחק.

מטרת התוכנית

התוכנית תקבל לוח משחק במצב נתון ושחקן שכרגע תורו, ותחשב איזה מבין המהלכים האפשריים במצב הנתון (לפי החוקיות של תרגיל בית 2) הוא המהלך הכי טוב לשחקן שמשחק כעת.
את הגדרת "מהלך טוב ביותר" נפרט בהמשך.

תזכורת לתיאור כללי המשחק

כללי המשחק הם ככללי המשחק המקורי אך מופשטים יותר.

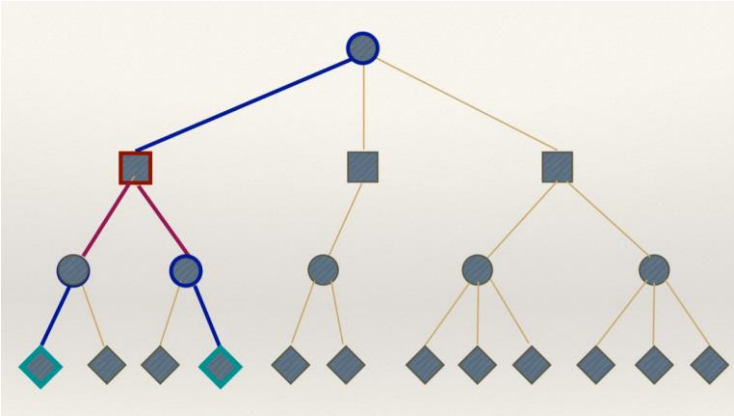
- לוח המשחק מורכב מ- 64 משבצות (8 X 8).
- לכל שחקן יש 12 אבנים, אחד עם אבנים בצבע שחור ואחד עם אבנים בצבע לבן.
- האבנים מונחות על המשבצות השחורות של הלוח וההתקדמות היא רק על משבצות אלה באלכסון.
- בתחילת המשחק אבני השחקן הראשון מונחות בשלוש השורות הראשונות של הלוח ואילו אבני השחקן השני מונחות באותו אופן בצד שלו.
- הלבן מבצע את המהלך הראשון, והשחור משיב עליו.
- **תנועה** : כל שחקן מניע בתורו אבן-משחק באלכסון, ממשבצת שחורה אחת למשבצת שחורה סמוכה בכיוון היריב. על המשבצת להיות פנויה מכלים, כלומר לכל אבן יש שתי אפשרויות תנועה – לכיוון היריב באלכסון ימינה ולכיוון היריב באלכסון שמאלה – וכל אחת מהן עשויה להיות חסומה.
- **אכילה (דילוג)** : מתבצע כאשר אבן משחק מונחת במשבצת סמוכה לאבן היריב, ומעבר לאבן היריב יש מקום פנוי.
- אין דילוגים מרובים, כלומר כאשר מבצעים דילוג נגמר התור ואי אפשר להמשיך.
- כאשר יש דילוג אפשרי, לא חובה לבצע אותו.
- אין מלכים במשחק, כלומר כשאבן משחק מגיעה לשורה האחרונה (שורה ראשונה של היריב) היא לא הופכת למלך אלא פשוט אי אפשר להזיז את האבן הזו לאורך שאר המשחק.
- המשחק יכול להסתיים בניצחון או בתיקו. ניצחון מושג אם מתקיים אחד מהבאים :
  - לשחקן היריב לא נותרו כלל כלים על הלוח.
  - לשחקנים אין אפשרות לבצע מהלך מאחר שכליהם חסומים ולירביב יש כמות קטנה יותר של כלים על הלוח.
- המקרה היחיד שמוגדר כתיקו הוא כאשר לשחקנים אין אפשרות לבצע מהלך ומספר הכלים של שני השחקנים על הלוח שווה.

הערה : חלק מההערות לא יהיו שימושיות בתרגיל זה (הן רלוונטיות לתרגילים הבאים).

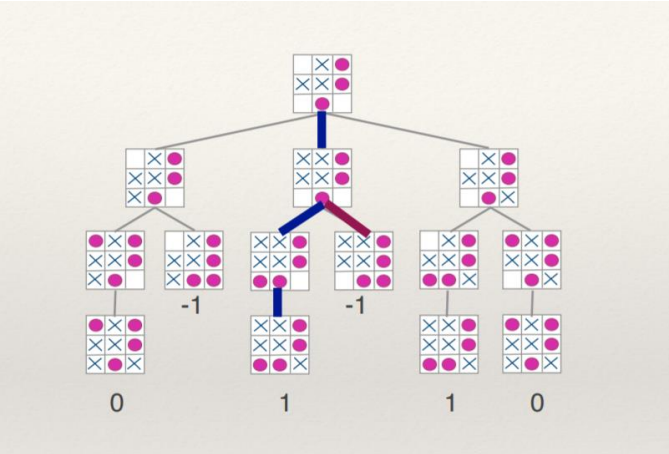
עצי משחק

בתרגיל זה נמצא עבור מצב נתון של משחק את הצעד הבא האופטימאלי לביצוע במשחק דמקה.
הטכניקה מוגדרת עבור משחק מבוסס שני שחקנים, כך שעבור מצב נוכחי אפשר לדעת בעזרת העץ מה הצעד האופטימאלי לביצוע בכל שלב בעזרת פונקציה שמוגדרת עבור מצב לוח כלשהו.
נגדיר את העץ ככה ששורש העץ – הוא המצב התחלתי שממנו אנו בודקים את המשחק שלנו ומה הצעד הכי טוב שנוכל לבצע כעת.
עבור כל צומת נגדיר שהצומת מייצג מצב כלשהו של לוח המשחק.
כלומר, העץ מייצג את המהלכים האפשריים מכל צומת, כאשר כל בן של צומת הוא מהלך אפשרי אחר.
נסתכל על דוגמה כללית:

עיגול - שחקן א'
ריבוע - שחקן ב'



העיגול בשורש (העיגול העליון בציור) מייצג את המצב ההתחלתי שממנו שחקן א' מתחיל. אנו בודקים כל צעד אפשרי מהמצב התחלתי ומייצגים אותו ע"י קשת שמובילה לאחד מבניו, המייצג מצב נוכחי של השחקן השני – הריבוע . מהשחקן הריבוע באותו אופן לכל צעד אפשרי ישנה קשת שמובילה הפעם לעיגול חזרה לשחקן הראשון וחוזר חלילה.
נסתכל על דוגמה ספציפית עבור המשחק איקס עיגול. במשחקים מורכבים יותר שבהם העץ יכול לגדול לממדים גדולים מדי מגדירים עומק בדיקה (שיוגדר באופן דומה אצלנו) ולפי העומק ניתן לדעת האם יש לחפש תוצאה מקסימלית או מינימלית.
בדוגמה זו כל צומת מייצג מצב של הלוח של איקס עיגול.



המצב ההתחלתי הוא שעכשיו תורו של איקס כמתואר בריבוע העליון בציור למעלה. ניתן לראות שעבור כל מיקום אפשרי בו השחקן של איקס יכול לשים X נוצר תת-עץ שממנו בודקים עבור עיגול את כל האפשרויות עבורו וכן הלאה . עבור העלים (לוחות שמייצגים משחק שנגמר), מוגדרות שלוש תוצאות אפשריות כתוצאת סיום המשחק $1, 0, -1$ שמוגדרת לפי השחקן שאיתו התחלנו. כלומר, אם נקבל בקצה של עץ תוצאה 1 אז איקס ניצח אם נקבל 1- אז עיגול ניצח ועבור 0 התוצאה היא תיקו .כמו שנאמר מקודם, כיוון שכדי להגיע למצב מסיים כזה נדרשים משאבי חישוב רבים, ניתן לקבוע עומק חישוב ולתת תוצאה גם למצב שאינו מסיים (לפי הגדרת מאפיינים כלשהם). למשל, ניתן להגדיר כי אנו מבצעים הערכה למצבים בעומק n (או מצבים מסיימים אם פגשנו כזה בעומק קטן מ-n).

ניתן להשתמש בעץ משחק כזה כדי למצוא מה המהלך שהכי כדאי לשחקן X לבצע באמצעות האלגוריתם הבא - בהינתן צומת בעץ node:

1. בכל צומת שנפגוש:
   a. אם node מייצג מצב סיום משחק **או** ש-node נמצא בעומק n, חשב את ה"תוצאה" עבור מצב זה של הלוח (כמו ה 1-/0/1 בשח-מט). נשים לב שחישוב התוצאה בד"כ **תלוי במי השחקן** שזה מצב הלוח בתורו.
   b. אחרת, אם node מייצג את מצב הלוח בתור של שחקן X, בדוק את ה"תוצאה" של כל הבנים של node והחזר את **המהלך עם התוצאה המקסימלית** (כך ש-X יבצע את המהלך שייתן לו את התוצאה הכי כדאית).
   c. אחרת, אם node מייצג את המצב הלוח בתור של היריב של X, בדוק את ה"תוצאה" של כל הבנים של node והחזר את **המהלך עם התוצאה המינימלית** (במקרה זה, אנו מניחים שהיריב יעשה את המהלך שהכי גרוע ל-X).

ובאופן יותר מפורט האלגוריתם מקבל כקלט:

- צומת בעץ node – שמייצגת לוח כלשהו של המשחק.
- עומק מקסימלי לחיפוש depth.
- שחקן נוכחי Player (כך שנקרא ליריב Adversary).

ופועל באופן שמתואר ע"י הפסאודו קוד הבא:

minmax (node , depth ,Player)

1. אם depth ==0 **או** nodeש מייצג לוח משחק הנמצא במצב סוף משחק (ניצחון\הפסד\תיקו)
   1.1. חשב את הערך של ה"תוצאה" לפי המצב הנוכחי של לוח המשחק שה node מייצג (בעזרת הפרמטר שקובע מי הוא השחקן הנוכחי)
2. אחרת , אם השחקן הנוכחי הוא השחקן Player
   2.1. ערך מיטבי = $-\infty$ (ערך הכי נמוך אפשרי)
   2.2. לכל בן של הצומת node בצע
      2.2.1. הפעל באופן רקורסיבי את האלגוריתם minimax עם הבן הנבדק ,הקטן את העומק ב1, ושלח כשחקן את adversary ושמור את ערך התוצאה במשתנה ערך נוכחי
      currValue = minimax(child,depth-1, Adversary)
      2.2.2. בדוק אם הערך הנוכחי גדול מהערך המיטבי ואם כן בצע עדכון
      bestValue =max(bestValue,currValue)
3. אחרת , אם השחקן הנוכחי הוא השחקן Adversary
   3.1. ערך מיטבי = $+\infty$ (ערך הכי גבוה אפשרי)
   3.2. לכל בן של הצומת node בצע
      3.2.1. הפעל באופן רקורסיבי את האלגוריתם minimax עם הבן הנבדק ,הקטן את העומק ב1,ושלח כשחקן את Player ושמור את ערך התוצאה במשתנה ערך נוכחי .
      currValue = minimax(child,depth-1, Player)
      3.2.2. בדוק אם הערך הנוכחי קטן מהערך המיטבי ואם כן בצע עדכון
      bestValue =min(bestValue,currValue)

להרחבה על האלגוריתם הזה (הנקרא MiniMax), ניתן לקרוא עליו עוד באינטרנט (למשל כאן)

קלט התכנית

התוכנית תקבל בתוויות הבאות:

- **Board** – תווית המציינת <u>מערך מלים</u> המייצג את לוח המשחק. המערך יהיה בגודל 8x8 מלים, ויאוחסן בזיכרון שורה-שורה (כמו מערכים דו-מימדיים בשפת C). כל מילה תכיל אחד משלושה ערכים:
  - 0 – מייצג תא ריק.
  - 1 – מייצג כלי לבן.
  - 2 – מייצג כלי שחור.
- **Player** – תווית המציינת <u>מילה</u> בזיכרון <u>שיכולה להכיל את הערכים הבאים:</u>
  - 1 – צריך לחשב את המהלך האופטימאלי עבור הכלים הלבנים.
  - 2 – צריך לחשב את המהלך האופטימאלי עבור הכלים השחורים.
- **Steps** – תווית המציינת <u>מילה</u> בזיכרון שמייצגת את כמות צעדי המשחק שעליכם לבדוק קדימה בעת חיפוש המהלך האופטימאלי.

<u>שימו לב</u>: הנחות לגבי תקינות הקלט מופיעות בסוף מסמך זה, תחת "הערות נוספות". אנא קראו אותן בעיון.

המשימה שלכם:

עליכם לכתוב תוכנית באסמבלי שתחשב עבור השחקן הנוכחי (שמוגדר ע"י התווית **Player**) את המהלך האופטימאלי שהא יכול לבצע בשלב זה ע"י הרצת כל המהלכים האפשריים ובדיקתם על פי הפרמטר **WP- Winning Parameter** באותה שיטה של עצי משחק שהוגדרה בעמוד קודם.
כל החישובים של פרמטר ההצלחה מחושבים בבסיס אוקטאלי .
הפרמטר מודד כמה המצב הנוכחי של הלוח הוא עדיף לשחקן כלשהו מוגדר :

#W = מספר הכלים הלבנים על הלוח
#B = מספר הכלים השחורים על הלוח

עבור השחקן השחור **WP** מוגדר כך :

$$WP_B = \#B - \#W$$

ובאופן דומה עבור השחקן הלבן **WP** מוגדר כך :

$$WP_W = \#W - \#B$$

בנוסף נגדיר עבור ניצחון של שחקן שה-WP עבור המצב ללא תלות בלוח הוא 20 ועבור הפסד הוא 20-.

שימו לב : בעת חישוב הפרמטר הנ"ל צריך להתחשב במי הוא השחקן הנוכחי. אם זה תור השחקן שעליו אנחנו בודקים אז יש לבחור בערך המקסימלי ואחרת נבחר בערך המינימלי.

הבדיקה שלכם תתבצע באופן הבא:
באותה הדרך שבה ראיתם על משחק האיקס עיגול, עליכם להשתמש באותה הטכניקה ולשחק את כל המהלכים האפשריים מהמצב הנתון שקיבלתם ועבור כל מצב לוח שמתקבל ממהלך שלכם לשחק את כל המהלכים של היריב על פי המצב החדש וחוזר חלילה לפי כמות הצעדים שמוגדרים בתווית **Steps.**
לדוגמא אם בתווית **Steps** מופיע הערך 5 עליכם לבדוק 5 צעדים קדימה את כל האופציות בכל אחד מהמצבים שניתן להגיע אליהם תוך חמש צעדים ולהחזיר את **WP** המיטבי ואיך ניתן להגיע אליו.

הערות חשובות:

- במידה ולפי הפרמטר ישנם שני מהלכים שהליכה בהם תניב את אותו מידת הצלחה לפי **WP** אז התוכנית תחזיר את אחד מהם ללא חשיבות.
- במידה וניתן להשיג ניצחון תוך כמות קטנה או שווה לכמות המוגדרת בתווית **Steps** הצעד האופטימאלי הוא צעד שיכול להוביל לניצחון הזה (במידה ויש כמה כאלה התוכנית תחזיר אחד מהם ללא חשיבות לסדר).
- **אין צורך במהלך התרגיל לבנות את עץ המשחק עצמו והוא נועד רק להמחשה של דרך הפתרון הרצויה**, רקורסיה.

פלט התוכנית

עליכם לדאוג כי בכתובת המצוינת

ע"י התווית **SrcPos** ירשם האינדקס בלוח שבו נמצא השחקן שחישבתם שצריך לזוז על פי המספור בתמונה בהמשך.

ע"י התווית **DstPos** ירשם האינדקס בלוח שאליו השחקן שחישבתם צריך לזוז כדי לקבל את הערך **WP** המיטבי שקיבתלם על פי המספור בתמונה בהמשך.

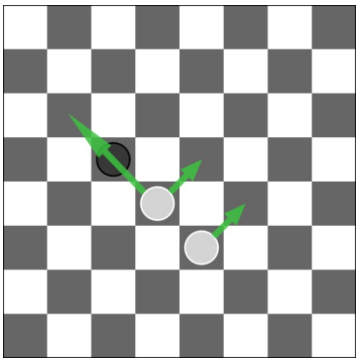
ולבסוף בכתובת המצוינת ע"י התווית **WinParam** - יופיע הפרמטר המיטבי שחישבתם– מקסימלי עבור Steps אי זוגי ומינימלי עבור Steps זוגי.

אופן מספור הלוח:

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

דוגמאות לקלט ופלט

דוגמה 1: עבור הקלט הבא:



```
Board:      .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 2, 0, 0, 0, 0, 0
            .word  0, 0, 0, 1, 0, 0, 0, 0
            .word  0, 0, 0, 0, 1, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
Player:     .word  1
Steps:      .word  2
```

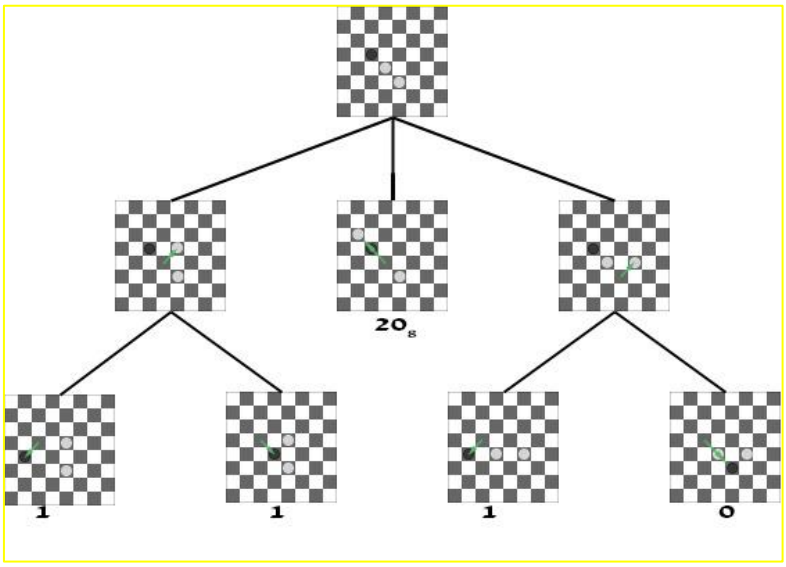
הפלט לאחר הרצת התוכנית הוא:

SrcPos = 35

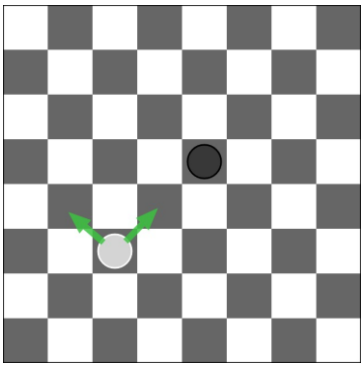
DstPos = 17

WinParam = 20

עץ המשחק המתאים לדוגמה:



דוגמה 2: עבור הקלט הבא:



```
Board:      .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 2, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 1, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
Player:     .word  1
Steps:      .word  1
```

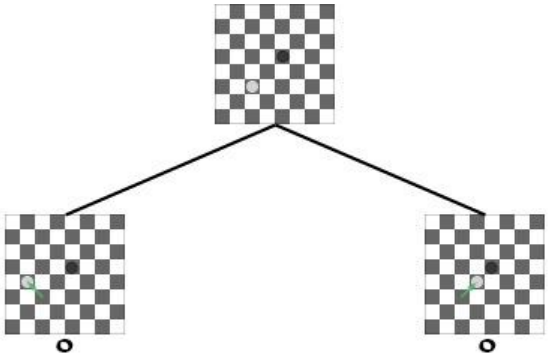
הפלט לאחר הרצת התוכנית הוא:

SrcPos = 42

DstPos = 33\35

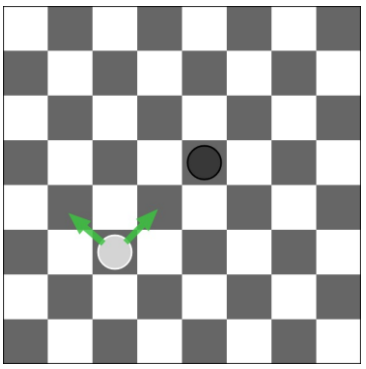
WinParam = 0

עץ המשחק המתאים לתמונה:



מכיוון שכמות הצעדים הנדרשת היא רק 1 אז אין חשיבות האם הצעד הוא ימינה באלכסון למשבצת 35 או שמאלה באלכסון למשבצת 33. נראה בדוגמא 3 מצב שונה:

דוגמה 3: עבור הקלט הבא (זהה לחלוטין לדוגמה קודמת מלבד מספר הצעדים):



```
Board:      .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 2, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 1, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
            .word  0, 0, 0, 0, 0, 0, 0, 0
Player:     .word  1
Steps:      .word  2
```

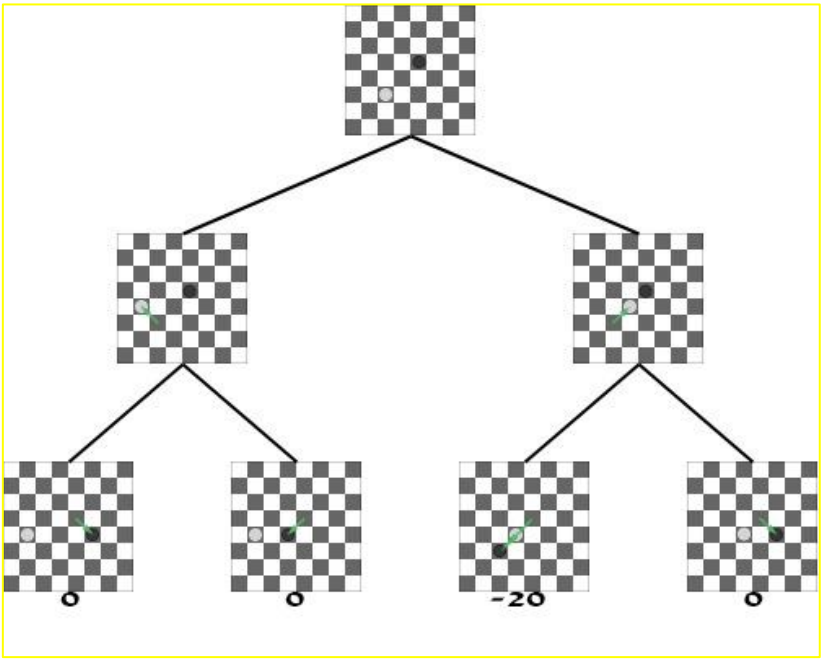
הפלט לאחר הרצת התוכנית הוא :

SrcPos = 42

DstPos = 33

WinParam = 0

עץ המשחק המתאים לתמונה:



הפעם המשבצת שאליה יזוז השחקן היא בהכרח 33 מכיוון שהתוכנה זיהתה שבמידה והוא יעבור ל35 הוא יאכל בתור הבא .

שגרות

בתוכנית שלכם **עליכם לממש** מספר שגרות שיפורטו להלן. לכל שיגרה מוסבר תפקידה והמנשק שלה (מהו אוגר הקישור, כיצד היא מקבלת ומחזירה פרמטרים וכדומה). הקפידו לממש את המנשק **במדויק**. אסור לשגרות להסתמך על ערכו של אף משתנה גלובאלי אלא אם נאמר אחרת.
שימו לב שיש **לתעד** כל אחת מהשגרות שאתם כותבים (גם כאלה שאינן ברשימה) כפי שמוסבר באתר הקורס.

שם השגרה	תפקיד השגרה	אוגר קישור	פרמטרים ושטח העברתם
calcWP	מקבלת את המצב הנוכחי של הלוח ואת השחקן שעכשיו תורו ומחשבת עבורו את **WP** לפי הגדרה	**pc**	**קלט:** הכתובת של התווית המכילה את המצב נוכחי מועברת במחסנית .המספר של השחקן הנוכחי (1 או 2) מועברת במחסנית (by value) **פלט:** החזר ברגיסטר r4 את **WP** המתאים
checkWin	מקבלת את המצב הנוכחי של הלוח ואת השחקן הנוכחי ומחזירה האם הוא במצב של ניצחון כעת	**pc**	**קלט:** הכתובת של התווית המכילה את המצב נוכחי מועברת במחסנית . המספר של השחקן הנוכחי (1 או 2) מועברת במחסנית (by value) **פלט:** במידה והלוח במצב של ניצחון עבור השחקן הנוכחי יוחזר 1 אחרת יוחזר 1-. מועבר במחסנית
minOrMax	הפונקציה מחזירה האם בשלב הנוכחי אנחנו מחפשים ערך מקסימלי או ערך מינמלי לפי הגדרות המשחק	**pc**	**קלט:** שחקן נוכחי ,יועבר במחסנית ,הכתובת של התווית שמכילה את השחקן שעבורנו אנו בודקים (התחלתי) יועבר במחסנית . **פלט:** תוצאה מוחזרת ב r4 . יוחזר 1 אם מחפשים ערך מקסימלי ו1- אם מחפשים ערך מינימלי .
movPiece	הפונקציה מקבל מקור ויעד של חייל על הלוח ומזיזה אותו ומעדכנת את הלוח בהתאם	**r5**	**קלט:** נקודת המוצא והיעד של החייל יועברו inline , כתובת התווית המכילה את הלוח הנוכחי תועבר במחסנית **פלט:** עדכון מצב הלוח הנוכחי לפי הפרמטרים הנתונים הלוח שמופיע בתווית שהועברה
getMaxWP	פונקציית המשחק הראשית שמחזירה את החייל שצריך לזוז ולאן ומה ערך	**pc**	**קלט:** הכתובת של התוויות שבהן צריך לרשום את הפלט יועברו במחסנית , המצב ההתחלתי,השחקן שעבורו אנו בודקים ועומק הבדיקה יעברו בשטח משותף בתוויות **Board** , **Steps** , **Player** **פלט:** החייל שצריך לזוז ולאן הוא צריך וערך ה**WP** יכתבו למקום שהועבר במחסנית

שימו לב:

i. אתם יכולים לממש כל שיגרת עזר או שגרת מעטפת בנוסף לשגרות שמפורטות למעלה.
ii. כל אחת מהשגרות הנ"ל יכולה להשתמש בכל אחת מהשגרות האחרות.
iii. למרות הדרישה לממש את כל השגרות למעלה, אין זה חובה להשתמש בשגרות אלה
iv. שימו לב להבדל בין מקומות שבהם נדרש להעביר by address לעומת by value

תהליך בדיקת נכונות התוכנית

כחלק מבדיקת התרגיל, תיבדק גם נכונות הריצה של התוכנית. תהליך הבדיקה נעשה על ידי הוספת הקלט (כלומר הוספת התוויות Board, Player) לסוף הקובץ אותו אתם מגישים, וכן הוספת תוויות המשמשות לפלט (התוויות של הפלט כגון **PieceToMove**), כל אלו בכתובות מעל 11000_8. לכן, אין להשתמש בכתובות מעל 11000_8 בכתיבת התוכנית. כמו כן, אין להגיש קובץ המכיל את הגדרות התוויות הנ"ל (שכן הגדרות אלו מוספות במהלך הבדיקה). אתם, כמובן, רשאים להוסיף תוויות אלו במהלך כתיבת התוכנית וניפוי השגיאות (debugging), אך, כאמור, אין להגיש את התוכנית שלכם עם הגדרת התוויות הנ"ל.
לצורך הבהרת עניין זה, יסופקו שני קבצים: ex3_test.txt ו- ex3_test.bat. הקובץ ex3_test.txt מכיל את ההגדרות של תוויות אלו, והקובץ ex3_test.bat הוא קובץ הרצה המשמש להוספת התוויות. עליכם לבצע את הפעולות הבאות לפני הגשת התרגיל:

1. יש לוודא כי שם הקובץ של התוכנית הוא ex3.s11,
2. להוריד את שני הקבצים (ex3_test.txt ו-ex3_test.bat) מהאתר לאותו המיקום בו נמצא קובץ התוכנית.
3. להריץ את הקובץ ex3_test.bat.
4. ייוצר קובץ חדש בשם ex3_temp.s11 המכיל את קוד התוכנית המקורי (מהקובץ ex3.s11) וכן את הגדרת התוויות (מהקובץ ex3_test.txt). יש לוודא כי עבור הקובץ החדש אין שגיאה בזמן תרגום וכי התוכנית מביאה לפלט הצפוי.
5. בכל אופן, יש להגיש את הקובץ ex3.s11.

שימו לב: לא יתקבלו ערעורים הקשורים בעניין הטכני הנ"ל.

הערות נוספות

1. ניתן להניח כי **הקלט תקין**, כלומר:
   - הלוח הוא בגודל 8x8.
   - הלוח מכיל רק את הערכים 0, 1, 2.
   - Player מכיל 1 או 2.
   - Steps מכיל מספר צעדים חיובי גדול ממש מאפס.
   - מצב המשחק הוא מצב חוקי, כלומר אפשר להגיע אליו על ידי מהלכים חוקיים.
2. התוכנית צריכה לפעול נכון עבור **כל** קלט תקין.
3. שימו לב לאותיות **גדולות/קטנות** בשימוש בכל **התוויות**.
4. התוכנית צריכה לרוץ על הסימולטור המסופק באתר הקורס.
5. **יש להקפיד על תיעוד פנימי וחיצוני של התוכנית.** יורדו נקודות בגין תיעוד לא מלא. קיים מסמך באתר הקורס תחת לשונית תרגילי הבית המסביר **כיצד יש לתעד**. תיעוד חיצוני יהיה **לכל היותר** 4-5 עמודים, ולא צריך לכלול את הקוד שלכם.
6. שאלות על התרגיל יש להפנות **ליותם זיו** בלבד.
7. **הגשות באיחור יש לתאם לפני מועד ההגשה.**
8. הגשה לתא הקורס: דף שער (נמצא באתר הקורס) + תשובות לחלק היבש (ללא דפי ההוראות של החלק הרטוב) + תיעוד חיצוני (**לכל היותר** 4-5 עמודים).
   הגשה אלקטרונית: קובץ הקוד ex3.s11 בלבד (הכולל בתוכו גם תיעוד פנימי).
9. **ההגשה בזוגות בלבד!**

עבודה נעימה!