

פרק 8 מחסנית ותור

1. הזמן הנדרש

8 שעות.

2. מטרות

1. היכרות עם שני סוגי אוספים כלליים : מחסנית ותור.
2. כתיבת מחלקה המיוצגת בעזרת שרשרת חוליות.
3. מימוש אוסף גנרי.

3. מבנה הפרק

א – המחסנית

- הציור שמופיע בפרק הוא הציור שמקובל לייצג בו את המחסנית. אך חשוב להדגיש כי אין קשר בין הציור למצב המחסנית בזיכרון. הציור מיועד להבנת אופן העבודה עם מחסנית. בהמשך, כאשר מדברים על ייצוג המערך, מבינים כי המחסנית יכולה להיות מיוצגת באמצעות מערך או שרשרת חוליות שאינם נראים כמו ציור המחסנית שלפניכם.
- פעולות ההכנסה וההוצאה הקיימות בכל אוסף של נתונים נקראות במחסנית : דחיפה ושליפה. אנו משתמשים בשמות אלה כיוון שאלה השמות המקובלים. כדי שהתלמידים יידעו שאלה אכן אותן פעולות רגילות הנעשות על אוספים, מדי פעם מוזכרות הפעולות בשמן הכללי : הכנסה והוצאה.

ב – שימושים במחסנית

- בכתיבת יישומים של אוספים, נקדים את השימוש באוספים להצגת ממשק מסודר שלהם, ונממש אותם. בדרך זו נפתח מוטיבציה ללמידת הנושא.
- מומלץ לפתוח את הנושא בהצגת הבעיות : היפוך קלט או בדיקת איזון סוגריים ולדון בפתרון שלהם. אם התלמידים מעלים פתרונות רקורסיביים, זו התחלה מצוינת. לאחר מכן ניתן לבקש מהם לפתור את הבעיה ללא רקורסיה (כל הפתרונות יהיו ברמה אלגוריתמית בלבד), ואז להציג את המחסנית ככלי לפתרון.
- בכיתות חזקות ניתן לפתח את הנושא של מחסנית זמן ריצה. מחסנית זמן ריצה היא יישום ממשי שעושה המחשב במחסנית פנימית משלו, כדי להיות מסוגל לעקוב אחר סדר ביצוע הפקודות ולעבוד נכון במצב שבו יש זימונים של פעולות, האחת מתוך האחרת. פקודות בתוכנית נשמרות בזיכרון ויש להן כתובות. כאשר תוכנית מתבצעת באופן רציף הפקודות מתבצעות לפי סדר עוקב. מה קורה כאשר פעולה מזמנת פעולה אחרת? לכאורה הריצה הסדרתית נקטעת, הביצוע עובר לשטח זיכרון אחר ולפקודות השמורות בו,

אך יש לזכור היטב ובמדויק מהיכן "קפצנו" ולאן אנחנו צריכים לחזור כדי להמשיך את רצף התוכנית לאחר תום הפעולה שזומנה. בדיוק לתכלית זו המחשב משתמש במחסנית זמן ריצה. ברגע הזימון של פעולה, המהדר מכניס למחסנית זמן הריצה את כתובת הפקודה הבאה אחרי פקודת הזימון, ששם תימשך הריצה של התוכנית כשהיא תשוב מהזימון.

מחסנית זמן ריצה בזימון פעולות רקורסיביות :

המערכת מסוגלת לזהות מצב שבו לא נכתב תנאי עצירה בפעולה רקורסיבית, נוצר מספר הולך וגדל של קריאות רקורסיביות כאשר מחסנית זמן הריצה מתמלאת. משום שהמחסנית מלאה, המערכת אינה מסוגלת עוד לבצע זימונים. התוכנית תיעצר בגלל חוסר מקום במחסנית זמן ריצה, כלומר תתקבל שגיאה בזמן ריצה : StackOverflow exception.

- נקודה מעניינת למחשבה: כיוון שרקורסיה נעזרת במחסנית, ניתן לעתים קרובות לפתור אותם תרגילים בעזרת רקורסיה או בעזרת מחסנית, שכן הם ממלאים תפקיד דומה בפתרון התרגילים. למשל אפשר לעשות פעולת "הפיכה" באמצעות שניהם (ראו את תרגיל 10, שבו הופכים מחסנית ללא מחסנית עזר).

- **בדיקת סוגריים:** שימו לב כי כשאנו מגדירים כי ביטוי חשבוני הוא תקין, איננו טוענים בהכרח שהוא תקין כביטוי חשבוני אלא רק שהוא מאוזן מבחינת סוגריים. בפרק הגדרנו אלגוריתם לפתרון המשימה של בדיקת תקינות סוגריים. בהמשך התלמידים יתבקשו לממש את האלגוריתם.

ג – ממשק המחלקה מחסנית

- בכיתות החלשות יותר כדאי ללמד קודם מחסנית שאינה גנרית. למשל CharStack ו-IntStack, ורק אז להגיע למחסנית הגנרית. זאת כדי להפריד בין שני התכנים: גנריות ורעיון המחסנית. בכיתות חזקות שהפנימו את הגנריות כבר בפרק הקודם, ניתן לכתוב מיד מחסנית גנרית.
- שימו לב כי זאת הפעם הראשונה שהתלמידים פוגשים מחלקה המייצגת אוסף גנרי (בפרק הקודם ראינו אמנם שרשרת חוליות גנריות אך השרשרת לא הייתה עטופה כמחלקה). חשוב להבין כמה נקודות הקשורות באוסף גנרי:

כל הפעולות בתוך המחלקה הגנרית פועלות על **T כטיפוס כללי ביותר**, כלומר טיפוס אשר אין לו שום התחייבות לממש פעולה מסוימת, פרט לפעולות המוגדרות במחלקה Object. לכן לא ניתן להניח כי לטיפוס זה קיימות פעולות מסוימות, פרט לאלה של Object. למשל, אם בתוך פעולה קיים משתנה x מטיפוס T, ניתן להפעיל עליו את פעולת equals(...) / equals(...), שכן פעולה זו מוגדרת ב-Object. אפשר גם להגדיר מחדש פעולה זו במחלקה הגנרית. לא חוקי להשתמש על עצמים מטיפוס T בפעולה אריתמטית כגון חיבור, או בפעולת השוואה בעזרת הסימן <. גם ניסיון להשתמש בפעולה המוגדרת במחלקה כלשהי פרט ל-Object ייכשל.

לכן, לא ניתן לכתוב פעולות פנימיות שדורשות התייחסות לטיפוסים בעלי תכונות מיוחדות (פעולות ייחודיות). בעזרת כלים מתקדמים הקיימים בשפות כמו ממשקים, ניתן להגדיר גם מחלקות גנריות היכולות להתייחס לטיפוסים שכאלה.

- השאלות הבאות הן דוגמאות לשאלות שאינן ניתנות לפתרון עקב בעיית הגריות שתוארה: "כתבו פעולה פנימית הסוכמת את ערכי המחסנית" – שכן ההנחה המונחת בבסיס השאלה היא שערכי המחסנית ניתנים לסכימה; וכן: "כתבו פעולה פנימית המחזירה את הערך המקסימלי השמור במחסנית" – שכן ההנחה היא שערכי המחסנית ניתנים להשוואה. במקום זאת ניתן לכתוב פעולות אלה כפעולות חיזוניות הפועלות על מחסנית מטיפוס מסוים, קונקרטי.

ד – שימוש בפעולות הממשק

בתוכנית המדפיסה את סדרת התווים שנקלטה בסדר הפוך, אנו משתמשים במחלקה Scanner המוכרת לתלמידים מלימודיהם הקודמים, אך מוסיפים שימוש מעניין במפריד (delimiter):

```
Scanner in = new Scanner(System.in);
in.useDelimiter("");
```

אחרי זימון הפעולה useDelimiter(""), כל התווים עד ה-enter נקראים כתווים בודדים ולא ברצף. ללא זימון הפעולה, הקלט כולו היה נקלט ברגע הקשת ה-enter, ולא ניתן להפריד אותו לתווים. עם ההכרזה על המפריד, מסומן סוף הקלט על ידי enter, אבל אז מתחיל עיבוד של הקלט על פי התווים.

הסימון של ה-enter שנקלט בקלט הוא:

```
while (ch != '\r')
```

ה – כתיבת המחלקה מחסנית

- בסעיף זה אנו רואים כי עבור אותו ממשק של מחסנית ניתן לכתוב מחלקות שונות. תופעה זו קיימת גם בתור. זהו מאפיין עיקרי של טיפוס נתונים מופשט: שינוי הייצוג של מחלקה אינו גורם לשינוי הממשק שלה וגם שומר על הגדרת הפעולות (למשל, pop שולפת את הערך האחרון שהוכנס למחסנית ועדיין לא נשלף). לתלמידים הלומדים את היחידה החמישית של תכנות מונחה עצמים, אפשר להסביר כי ניתן להגדיר את ממשק המחסנית כ-interface, וכך שתי המחלקות יממשו אותו. כפי שנראה בהמשך, רשימה ועץ בינרי, כפי שהם מוגדרים ביחידה זו, אינם טיפוסים נתונים מופשטים.
- ייצוג מחסנית באמצעות מערך בג'אווה:

אתחול המערך arr נעשה בפעולה הבונה. כיוון שהמערך המשמש לאחסון איברי המחסנית הוא מערך גנרי, אתחולו נעשה באופן מיוחד:

```
public Stack()
{
    this.arr = (T[]) new Object[STACK_MAX_SIZE];
    this.top = -1;
}
```

הסבר: על פי דרישות השפה, מערך גנרי מוגדר בשני שלבים. בשלב ראשון, מוגדר מערך שאיבריו הם מטיפוס Object. בשלב שני, מתבצעת המרה של טיפוס הערכים שבמערך לטיפוס הגנרי T (השימוש בשני שלבים אלה נדרש בגלל אופן המימוש של גנריות בג'אווה).

כאשר מגדירים טיפוס קונקרטי עבור T ובונים מחסנית מטיפוס זה בעזרת הפעולה:

```
new Stack<String> ()
```

אזי הטיפוס String מחליף את T, בכל מקום שהוא מופיע. המתכנתים אינם רואים את ההחלפה. ההחלפה נעשית בפרט בפעולה הבונה, כלומר: ההמרה (T[]) מוחלפת בהמרה (String[]), ולכן במערכת נרשם כי בעצם שנוצר, המערך מחזיק הפניות לעצמים מטיפוס String.

חשוב לציין שתהליך ההגדרה של מערך גנרי פשוט יותר ב-C# מבחינת הכתיבה שלו. ההבדל אינו מייצג רשלנות של מתכנני ג'אווה, אלא השקפות שונות על תכנון שפה, אך ההסבר לכך דורש הבנה לעומק של מימושי הגנריות בשתי השפות, ואין מקומו כאן.

- ייצוג מחסנית באמצעות שרשרת חוליות:

אחרי הצגת הייצוג מומלץ לא להגיד לתלמידים כי ההכנסה מתבצעת לתחילת שרשרת החוליות, אלא לפתח דיון על אופן ההכנסה. לשאול את התלמידים לאן לדעתם כדאי להכניס, ולדון ביעילות הפעולות שיציעו. ייתכן שההכנסה לראש השרשרת תיראה להם לא טבעית (שהרי הם רגילים לעבוד עם מערך שבו ההכנסה מתבצעת לסוף).

בסעיף זו אנחנו מראים שעבור אותו ייצוג ממש (תכונה אחת המפנה לחוליה) ניתן לממש את המחסנית בשני אופנים שונים (תלוי איפה נקבע ראש המחסנית). אלה הם שני ייצוגים שונים אף שברמת קביעת התכונות אין שום הבדל ביניהם. מכאן שיש להסביר לתלמידים כי לצד קביעת הייצוג (הגדרת התכונות), יש להגדיר במפורש כחלק מהייצוג עצמו, היכן בדיוק מתבצעות הפעולות על המחסנית. פעם המחסנית מיוצגת על ידי שרשרת חוליות והפעילות כולה מתבצעת בראש השרשרת, בעוד שבייצוג האחר המחסנית מיוצגת גם כן על ידי שרשרת חוליות, אך הפעילות מתבצעת בקצה האחר של השרשרת.

תשובות לשאלות המחשבה המופיעות בסעיף זה:

? כתבו את המחלקה Stack<T> כאשר היא מיוצגת על ידי מערך.

תשובה:

ניתן להחליט שלא לבקש מהתלמידים לבצע את המימוש בפועל, כדי לא להסתבך בג'אווה עם הגדרת מערך גנרי וכדי לחסוך בזמן באופן כללי. עם זאת, חשוב שהנקודות העיקריות בצורת ייצוג זו יידונו, ושהתלמידים יבינו שיש דרך לייצג את המחסנית באופן שכזה. כאשר ייצוג המחלקה ישתנה בלי שהדבר ישפיע על הממשק, התלמידים גם יבינו את רעיון טיפוס הנתונים המופשט.

? הסבירו מדוע אין בקוד זה שום בעיה בהכנסת חוליה למקום הראשון בשרשרת.

תשובה:

בשרשרת חוליות הראינו כי קיימת בעיה להכניס את החוליה הראשונה ולהוציאה בתוך פעולות המקבלות את ההפניה לשרשרת החוליות כפרמטר. הבעיה קיימת כיוון שלא ניתן לשנות את המשתנה המכיל הפניה לשרשרת החוליות (ההפניה לחוליה הראשונה), מתוך פעולה שאליה היא נשלחת, שהרי המשתנה אינו מוכר לפעולה. כאשר שרשרת החוליות היא תכונה פנימית של אוסף, בעיה זו אינה קיימת. המשתנה היחיד המכיל הפניה לחוליה הראשונה הוא תכונה של המחלקה. משתנה זה מוכר לפעולות של המחלקה, ולכן הן יכולות לשנותו – התכונה מעודכנת ישירות בפעולות האוסף.

הערה: כפי שהסברנו עכשיו, כאשר עוטפים שרשרת חוליות במחלקה, כלומר מגדירים מחלקה שהשרשרת תכונה פנימית פרטית שלה, הבעיות של הכנסה או הוצאה למקום הראשון או ממנו, על ידי פעולה - נעלמות. באופן דומה, שרשרת מטבעה אינה יכולה להיות ריקה. אך כאשר במחלקת אוסף ייצוג האוסף נעשה על ידי תכונה פנימית שערכה הוא שרשרת, אפשר להחליט כי כאשר ערך התכונה הוא `null` פירוש הדבר הוא שהאוסף ריק. כך אנו מתגברים גם על הבעיה של אי קיום שרשרת ריקה.

1 – יעילות פעולות המחסנית

אם מגדירים תכונה נוספת המפנה לסוף שרשרת, ניתן לשפר את יעילות פעולת ההכנסה למחסנית אם ההוספה מתבצעת בסוף שרשרת החוליות. אך במקרה זה, אחרי שמוציאים איבר מהמחסנית, יש צורך לעדכן את התכונה הזו. עדכון התכונה מחייב סריקה של שרשרת החוליות עד שמגיעים לאיבר האחרון, כלומר יעילותו היא $O(n)$.

סעיף זה מדבר על יעילות שמשנתנה בין מימושים שונים על סמך ייצוגים שונים. צריך לחזור ולהדגיש שתחת אותו ייצוג ייתכנו מימושים שונים שיעילותיהם שונות. בכל מצב שהתלמיד נדרש לייצג מחלקה, עליו להגדיר את הייצוג ואת המימוש המתאימה לדרך ייצוג זו. גם חישובי יעילות ניתן לעשות רק לאחר הגדרת ייצוג מלאה שכזו. את הרעיון הזה פגשנו בעבר, למשל כאשר עסקנו במחלקה `StudentList` והתלבטנו עבור אותו ייצוג אם להחזיק את הרשימה ממוינת כל הזמן, או למיין אותה רק לקראת הדפסה, דבר ששינה את יעילות הפעולות.

2 – פעולות נוספות

- יש פעולות חיצוניות שניתן היה להגדיר כגנריות כי הן לא נוגעות בערכי העצמים, ובכל זאת למען הפשטות והאחידות, כל הפעולות החיצוניות ביחידת הלימוד הנוכחית מקבלות כפרמטרים רק טיפוסים קונקרטיים.
- כזכור, אנו מניחים כהנחה סמויה, ואיננו חוזרים ומציינים אותה, כי פרמטר מטיפוס עצם לא יכול להיות `null`.

- בפעולה הסופרת איברים במחסנית יש הבדל מהותי-לשוני בין הפעולה החיצונית לפעולה הפנימית: בפעולה הפנימית סופרים חוליות ובפעולה החיצונית סופרים איברים במחסנית, שכן אין שום גישה לייצוג המחסנית.
- פעולות חיצוניות סטטיות גנריות. נושא זה איננו נכלל בתוכנית הלימודים, אך עלול להתעורר אצל הלומד, ולכן אנו מרחיבים מעט. נכתוב פעולה חיצונית המקבלת מחסנית ובודקת אם יש בה לפחות שלושה איברים. נניח שהפעולה נמצאת במחלקת השירות StackUtil. האפשרות הראשונה שנבחר היא להפוך את המחלקה לגנרית:

```
public class StackUtil<T>
{
    public static boolean hasThree (Stack<T> stack) {...}
}
```

כאן, הפעולה היא פעולה פנימית של מחלקה גנרית, ולכן מותר להשתמש בה בפרמטר הטיפוס הגנרי שהוכרז בכותרת המחלקה. אך המהדר ידחה את ההגדרה הזאת. נזכור שבזמן יצירת עצם מטיפוס StackUtil הטיפוס T יאותחל להיות טיפוס ספציפי כלשהו. כיוון שפעולה סטטית יכולה להיות מזומנת לפני שנוצר עצם כלשהו, אסור להשתמש בה בפרמטר גנרי של המחלקה.

האפשרות שנשארת לנו היא להגדיר את הפעולה עצמה כפעולה גנרית, שבכותרתה מוכרז פרמטר טיפוס גנרי, בעזרת התחביר הזה:

```
public class StackUtil
{
    public static <T> boolean hasThree (Stack<T> stack) {...}
}
```

כאן, המופע הראשון של <T> מכריז על T כפרמטר טיפוס גנרי של הפעולה. המופע השני הוא שימוש בו. בגלל בעיות שונות בשימוש בפעולות גנריות, ג'אוה מציעה גם מנגנון נוסף: תחביר חילופי שמשמעותו היא שהפעולה מקבלת מחסנית מטיפוס כלשהו (שימו לב שאפשרות זו אינה קיימת בסישרפ):

```
public class StackUtil
{
    public static boolean hasThree (Stack<?> stack) {...}
}
```

- הפעולה המחזירה את מספר האיברים השמורים במחסנית וכן הפעולה הסוכמת את האיברים שבמחסנית אינן טבעיות ומתאימות למבנה המחסנית. אפשר להוסיף פעולה כגון ספירת מספר האיברים לממשק הבסיסי, אם יש בה שימוש נרחב, וכמובן רק בתנאי שאינה משתמשת בפעולות ייחודיות של טיפוס הערך. (לכן, ספירת מספר האיברים יכולה להיכתב כפעולה פנימית, אך פעולת הסכימה אינה יכולה). מטרתנו כאן, בכל אופן, היא בעיקר לדון בהבדלים בין פעולות פנימיות וחיצוניות ולהמחיש אותם.

ח – המחסנית: טיפוס נתונים מופשט

המחסנית, כמו המחלקה StudentList, היא טיפוס נתונים מופשט. לעומת זאת, הרשימה, שהתלמידים יראו בהמשך, אינה טיפוס נתונים מופשט. אנו מזכירים שוב ושוב מה הוא טיפוס נתונים מופשט כדי שנוכל להראות מהם החסרונות של הרשימה, כפי שהיא מוגדרת אצלנו, כאשר איננו יכולים להגדירה כטיפוס נתונים מופשט בכלים העומדים לרשותנו.

ט – התור

התור אינו שונה באופן משמעותי ממחסנית, אך חשוב מאוד שהתלמידים יבינו את הפרוטוקול שהתור מתנהל לפיו ויממשו את המחלקה על פי הייצוגים המתאימים.

4. תרגילים

פתרונות לכל התרגילים מופיעים באתר המרכז להוראת המדעים, האוניברסיטה העברית בירושלים ונגישים רק לציבור המורים:

http://sites.huji.ac.il/science/unit4_2007/

התרגילים ממוינים לשלושה חלקים:

- א. תרגילי מעקב – לתרגול פעולות הממשקים של מחסנית ותור: תרגילים 1–8.
- ב. תרגילי מימוש – לתרגול כתיבת פעולות (פנימיות וחיצוניות) על מחסנית ותור: תרגילים 9–16.
- ג. תרגילי שימוש במחסנית ותור לפתרון בעיות: תרגילים 17–21 וכן דפי העבודה 1–2.

ניתן לחזור לתרגילים שנפתרו בעבר ולבקש מהתלמידים לפתור אותם שנית בעזרת מחסניות. לדוגמה, שאלות 9 ו-12 מפרק 7 – ייצוג אוספים, ניתנות לפתרון מעניין בעזרת מחסניות, שאליהן מוכנסות שרשרות הנתונים מסופן לתחילתן.

שאלה 1

מעקב אחר פעולות המחסנית.
אפשר לתת כשיעורי בית.

שאלה 2

מעקב אחר פעולות התור.
אפשר לתת כשיעורי בית.

שאלה 3

מעקב אחר פעולות המחסנית.
אפשר לתת כשיעורי בית.

שאלה 4

מעקב אחר פעולות המחסנית.
מומלץ לפתור בכיתה.

שאלה 5

מעקב אחר פעולות התור.
מומלץ לפתור בכיתה.
ככיתות טובות ניתן להפוך את השאלה, ולתת להם לכתוב את הקוד שמבצע את המטלה הזו.
נוסח השאלה לקוח מתוך ויקיפדיה.

שאלה 6

מעקב אחר פעולות המחסנית.
אפשר לתת כשיעורי בית.

שאלה 7

מעקב אחר פעולות המחסנית.
אפשר לתת כשיעורי בית.

שאלה 8

מעקב אחר פעולה רקורסיבית הפועלת על תור.
מומלץ לפתור בכיתה.

שאלה 9

כתיבת פעולה הפועלת על תור.
מומלץ לפתור בכיתה.

שאלה 10

כתיבת פעולה רקורסיבית הפועלת על המחסנית.
מומלץ לפתור בכיתה.

שאלה 11

כתיבת פעולה הפועלת על תור.
ניתן לתת כשיעורי בית.

שאלה 12

כתיבת פעולה הפועלת על תור.
התרגיל מוגבל לתור של תווים, אף שאין סיבה ממשית להגבלה זו. הכוונה היא שהתלמיד לא יידרש לבצע העתקה עמוקה של העצם.
ניתן לבצע את ההעתקה גם ללא תור עזר, באמצעות הרעיון שהוצג בתרגיל 9 (הצבת null בסוף התור). ראו בקובץ הפתרונות.
פתרון שגוי שתלמידים עלולים להציע הוא פתרון המבצע העתקה על ידי השמה של הפניות (כדאי לדון על כך עם התלמידים).
מומלץ לפתור בכיתה או לתת כשיעורי בית.

שאלה 13

כתיבת פעולה הפועלת על מחסנית.
ניתן לתת כשיעורי בית.

שאלה 14

פעולה הפועלת על מחסנית.
ניתן לתת כשיעורי בית.

שאלה 15

פעולה הפועלת על תור ונעזרת במחסנית עזר כדי להפוך את האיברים.
ניתן לתת כשיעורי בית.

שאלה 16

פעולה הפועלת על מערך של מחסנית.
מומלץ לעשות בכיתה.

שאלה 17

פעולה הפותרת בעיה בעזרת מחסנית.
מומלץ לעשות בכיתה.

שאלה 18

פעולה הבודקת תקינות סוגריים לפי האלגוריתם בפרק.
מומלץ לעשות בכיתה.

שאלות 19–21

פעולות העוסקות בהפיכת מחרוזות על פי כללים מסוימים.
מומלץ לפתור אחד מהם בכיתה ואת היתר לתת כשיעורי בית.
בתרגילים אלה צריכים לקלוט סדרת תווים. קליטת תו אחר תו יכולה להתבצע בג'אווה על ידי שימוש ב-delimiter, כפי שמודגם בדוגמה ד.2. בפרק ובמדריך למורה.
ניתן לפתור תרגילים אלה גם ללא שימוש במחסנית (והתלמידים מוזמנים לנסות). אך השימוש במחסנית קל יותר ונוח יותר.

דף עבודה מספר 1 – מיון בסיס

דף העבודה עוסק במערך של תורים המממש מיון בסיס, Radix sort.
מומלץ להתחיל בכיתה ולסיים כשיעורי בית.

בכיתות שעלולות להתקשות בפתרון התרגיל, מומלץ לחלק את הפתרון ולעקוב אחריו יחד עם התלמידים.

דף עבודה מספר 2 – מגדלי האנוי

יש לדון עם התלמידים על הייצוג. ניתן לתת להם להתחיל לכתוב את הפתרון עם 3 מחסניות, עד שבפעולה moveDisc הם יגלו שכדאי להם יותר להשתמש במערך של מחסניות.
יש לשים לב להבדל בין האינדקסים במערך (שערכיהם נעים בין 0 ל-2) לבין האינדקסים של המוטות (הנעים בין 1 ל-3).