

פרק 7 ייצוג אוספים

1. זמן נדרש

12 שעות.

2. מטרות

1. היכרות עם חוליה שאינה גנרית (IntNode)
2. טיפול בשרשרת חוליות: הוצאה, הכנסה, סריקה ופעולות נוספות על השרשרת.
3. היכרות עם חוליה גנרית.
4. שרשרת חוליות כמבנה רקורסיבי.
5. שרשרת חוליות לייצוג של אוסף.
6. היכרות עם מבנים נוספים מבוססי חוליות.

3. מבוא כללי

פרק זה מכיל את רעיונות היסוד של היחידה כולה. הפרק מעביר את התלמיד מהיחידות הראשונות, יסודות, העוסקות באלגוריתמיקה בסיסית וטיפול בכמויות נתונים מצומצמות לטיפול בכמויות נתונים גדולות המאפיינות מערכות תוכנה. הפרק מציג את המוטיבציה לטיפול באוספים, סוקר את הכלים העומדים לרשות התלמיד בשלב זה של לימודיו לאחסון האוספים וטיפול בהם, תוך חידוד מגבלותיהם. מכאן נבנית הגישה של מבני חוליות שתשמש לצורך ייצוג כל הטיפוסים והמבנים שיוצגו בהמשך היחידה.

הבנה מעמיקה של כל הנושאים המוצגים בפרק תביא לכך שמרבית פרקי ההמשך יילמדו כהדגמה של היסודות שהונחו בפרק, תוך שהם מחדדים ומחדשים תובנות על בסיס מה שנלמד בפרק. לכן מוקדש לפרק זמן הוראה ממושך יחסית. המדריך שלהלן מפרט את שלבי ההוראה, ומומלץ מאוד שלא לקצר ולדלג בפרק זה על שום נושא וסעיף.

פרק זה הוא הבסיס לכל הפרקים הבאים, ומכיל רעיונות יסוד של היחידה. ככזה יש ללמדו בעיון ובשקידה.

4. מבנה הפרק

מבוא

המבוא עוסק באוספים באופן כללי, ובחירויות המערך כמבנה לייצוג אוסף. המבוא מציג את המוטיבציה ללמידת הנושא החדש – חוליה ומבני חוליות, כאלטרנטיבה למערך לצורך ייצוג אוספים.

א – החוליה, אבן יסוד למבנה נתונים דינמי

1. חוליה המכילה מספר שלם (IntNode)

- היכרות ראשונית עם החוליה. בחרנו להציג תחילה את החוליה הלא גנרית, כדי להתמקד ברעיון של החוליה. רק לאחר הבנת הרעיון והתחושה שמיותר להגדיר חוליה לכל טיפוס ערך, יהיה מקום להציג את רעיון הגנריות באופן טבעי וכהמשך נדרש לצורך המובן (כבר) בחוליה.
- הפעולה toString() / ToString() מדפיסה רק את ערך החוליה (ללא החוליה העוקבת). זאת, כיוון שהחוליה הבאה, אם היא קיימת, היא חלק ממבנה הבנוי מחוליות. הצגת מבנה כזה אינה שייכת ל-toString() / ToString() של חוליה בודדת. ההפניה לחוליה הבאה הקיימת בחוליה שייכת למבנה של החוליה, במובן זה שהיא מאפשרת לחבר אליה חוליה נוספת, אך היא אינה חלק מהערכים שבחוליה.. יש להפנות את תשומת לב התלמידים לכך שמדפיסים רק את הערך (שכן הם רגילים להדפיס את כל התכונות).
- הממשק המוצג עבור IntNode אינו ממשק סופי ומחייב (כיוון שאינו גנרי) אלא משמש אותנו כשלב בהוראה. כדי להדגיש זאת הוא ממוסגר בטבלה שקוויה גליים בהירים, להבדיל מהטבלאות של הממשקים הסופיים הממוסגרות במסגרת אחידה וכהה.

2. א. שרשרת חוליות

- הרעיון של שרשרת חוליות מוכר לתלמידים מהפרק הקודם. הם כבר יצרו שרשרות של עצמים שכולם מאותו הטיפוס.
- שימו לב כי בניית שרשרת, הכנסה, הוצאה וסריקה אינן מוצגות כאן כפעולות אלא נכתבות כקטעי קוד בתוך המחלקה הראשית, וזאת כדי שהתלמידים יבינו את הלוגיקה של המימוש. אולי קשה במראה עין להתרשם שכך הם פני הדברים, ולכן כדאי להעיר את תשומת לב התלמידים לכך. הדיון כיצד לממש משימות כגון אלו כפעולות מתקיים בסעיף הבא.
- דוגמת ההכנסה שבפרק מתבצעת במקום השני בשרשרת החוליות: ההפניה לחוליה הראשונה כבר נתונה, ונוח לבצע את ההכנסה אחריה. כדאי להראות לתלמידים שניתן להכניס גם למקום אחר בשרשרת, פרט למקום הראשון. כדי לעשות זאת יש לעמוד על החוליה שאחריה רוצים להכניס את החוליה החדשה. את הדיון בהכנסה למקום הראשון כדאי לדחות לסעיף הבא.
- כדאי למורה לשאול את התלמידים איך הם היו מבצעים את ההכנסה. אם הם יצינו כי קודם היו משנים את ה-next של pos, כך שיצביע אל החוליה החדשה, יש להראות להם בציור כי דבר כזה

היה מנתק את יתר החוליות. מכאן יובן מדוע יש לחבר את החוליה החדשה ליתר החוליות, ורק לאחר מכן לשנות את ההפניה לחוליה הקודמת כך שתצביע לחוליה החדשה.

- שימו לב, בסעיף המדבר על מעבר על שרשרת החוליות מתחילים לקרוא להפניה הראשונה chain. שם זה מדגיש את העובדה כי החוליה מחזיקה בשרשרת. השם הזה נוח במיוחד בסעיף הבא, כאשר מדברים על שרשרת חוליות המועברת כפרמטר. כמו כן להפניה משמשת אותנו לסריקה אנו קוראים pos (מהמילה position), זאת כדי להדגיש כי הפניה זו משמשת אותנו כמציין מקום.
- במעבר על שרשרת החוליות מוצגת "תבנית של קוד", כדי להראות את הדרך הכללית לסריקה בלי להתייחס לביצוע מסוים על ערך החוליה. אחרי שהתלמידים יתרגלו לתבנית של סריקה, יש לציין בפניהם כי לעתים קרובות יחולו שינויים בתבנית, אך הרעיון יישאר דומה. למשל, יהיו מקרים שבהם נוח יהיה לבדוק את החוליה הבאה ולא את הנוכחית. לכן לדוגמה ניתן לכתוב את פעולת הסכימה גם באופן הזה:

בג'אווה:

```
IntNode pos = chain;
int sum = pos.getInfo();
while (pos.getNext() != null)
{
    pos = pos.getNext();
    sum = sum + pos.getInfo();           // סיכום ערכי החוליות
}
```

בסישרפ:

```
IntNode pos = chain;
int sum = pos.GetInfo();
while (pos.GetNext() != null)
{
    pos = pos.GetNext();
    sum = sum + pos.GetInfo();         // סיכום ערכי החוליות
}
```

במקרים רבים נרצה לסרוק את השרשרת באמצעות בדיקה של החוליה הבאה: האם היא קיימת, האם הערך שבה מקיים תנאי מסוים. נעשה זאת על ידי שימוש ב-pos.getNext() / pos.GetNext(). למשל אם ברצוננו למצוא חוליה המיועדת למחיקה, רצוי לעצור את הסריקה בחוליה שלפני החוליה האמורה להימחק.

תשובות לשאלות המחשבה המופיעות בסעיף זה :

? כתבו קטע תוכנית המוצא את המספר הגדול ביותר בשרשרת חוליות המוחזקת על ידי chain.

תשובה :

בג'אווה :

```
IntNode pos = chain;
int max = pos.getInfo();

while (pos.getNext() != null)
{
    pos = pos.getNext();
    if (pos.getInfo() > max)
        max = pos.getInfo();
}
```

בסישרפ :

```
IntNode pos = chain;
int max = pos.GetInfo();

while (pos.GetNext() != null)
{
    pos = pos.GetNext();
    if (pos.GetInfo() > max)
        max = pos.GetInfo();
}
```

3.א. שרשרת חוליות כפרמטר לפעולות

- סעיף זה ממשיך לעסוק באותה הדוגמה שהופיעה בסעיף הקודם: סכום של שרשרת חוליות. החידוש של הסעיף הוא בכך שכאשר ההפניה עוברת כפרמטר, בתוך הפעולה נמצא העותק של ההפניה לחוליה הראשונה. כלומר מחוץ לפעולה קיימת הפניה לחוליה ראשונה הנקראת myChain, ובתוך הפעולה קיים עותק של אותה הפניה (כלומר הפניה נוספת הפונה לחוליה הראשונה) הנקרא chain. שינוי ההפניה בגוף הפעולה, אינו משפיע על ההפניה myChain. נקודה זו היא רגישה וחשובה מאוד. גם אם היינו רוצים לשנות את ההפניה myChain מתוך הפעולה, לא היינו מצליחים. המחשה לעניין זה באה לידי ביטוי בצבע החלש של myChain כאשר נמצאים בטווח הפעולה עצמה ו-myChain למעשה אינו מוכר ואינו ניתן לשינוי.
- בפעולה: `getChainSum(IntNode chain) / GetChainSum(IntNode chain)` בכוונה לא הקצאנו עוד משתנה לביצוע הסריקה, כדי להראות שאף שהמשתנה chain שהתקבל כפרמטר עובר שינוי, המשתנה שפונה לחוליה הראשונה מחוץ לפעולה אינו משתנה. בפעולה זו נצמדנו לתבנית שהצגנו קודם וסרקנו את השרשרת כל עוד chain לא שווה null. יש לשים לב כי אם chain הוא null בעת זימון הפעולה (מצב שאינו תקין ושאותו השארנו לטיפולו של מי שמזמן את הפעולה), אזי הפעולה תחזיר 0.

- יש לשים לב כי אם chain הוא null בעת זימון הפעולה (מצב שאינו תקין ושאותו השארנו לטיפולו של מי שמזמן את הפעולה), אזי הפעולה תחזיר 0.
- בפעולה המוצאת ערך מקסימלי בקטע של שרשרת חוליות, יש לשים לב כי איננו מעבירים את שרשרת החוליות כפרמטר. העברת המקום כפרמטר מספיקה, שכן המקום מחזיק גם את הקטע הרצוי של שרשרת החוליות.
- הפסקה דנה בפעולות המקבלות שרשרת חוליות כפרמטר. בגוף פעולות אלה אי אפשר לשנות משתנה הנמצא מחוץ לפעולה ומכיל הפניה לחוליה הראשונה.. למרות זאת ובניגוד לרושם שעלול להתקבל מהדיון בסעיף זה, ניתן להכניס ולהוציא את החוליה הראשונה בצורות שונות. למשל באופן הזה:
כדי להוציא את החוליה הראשונה ניתן להחליף את הערכים השמורים בחוליה הראשונה עם אלה השמורים בחוליה השנייה, ולהוציא את החוליה השנייה.
כדי להכניס חוליה ראשונה לשרשרת ניתן לבצע תהליך הפוך: להכניס את החוליה במקום השני, ואז להחליף בין הערכים שבחוליה הראשונה לאלה שבחוליה השנייה.
תלמידים יכולים להעיר על כך שיש פתרונות עוקפים לבעיה, כגון הפתרון שהצגנו, אך מטרתנו היא להגיע משרשרת החוליות אל המחלקה העוטפת: רשימה. לכן אין לנו צורך בפתרונות עוקפים אלה. די בהבנת הבעיה הבסיסית כדי שמהלך הלימוד יתקדם כראוי.
- כל המשימות ממומשות כאן כפעולות חיצוניות לשרשרת. תלמידים יכולים לתהות האם אפשר לממשן גם כפעולות של החוליה. התשובה היא שטכנית הדבר אמנם אפשרי, אך זו גישה לא טובה. כיוון שהפעולות מתבצעות על אוסף החוליות, אין זה הגיוני שבמחלקה חוליה, המייצגת חוליה בודדת, יהיו פעולות אשר מתבצעות על אוסף החוליות (למשל הכנסה במקום מסוים בשרשרת). כאשר נעסוק במחלקות העוטפות שרשרות של חוליות, יהיה סביר לממש פעולות מסוג זה כפעולות פנימיות של מחלקות אלה.

תשובות לשאלות המחשבה המופיעות בסעיף זה:

? הסבירו מדוע הבעיה שתיארנו אינה מתעוררת אם הכנסת הערך מתבצעת במקום שאינו הראשון בשרשרת.

תשובה:

כאשר הכנסת הערך מתבצעת במקום שאינו ראשון בשרשרת, אזי החוליה הראשונה אינה משתנה, וכל ההפניות אליה, כולל זאת שבמשתנה החיצוני, נשארות תקפות. הבעיה מתעוררת אך ורק כאשר יש צורך לשנות את החוליה הראשונה, ובהתאם את ההפניות אליה. בתוך הפעולה אנו מחזיקים עותק של ההפניה, אבל עותק אחר קיים במשתנה חיצוני לפעולה, ואותו אין אפשרות לשנות בפעולה.

? נתונה שרשרת חוליות המוחזקת על ידי myChain.

א. כתבו את הזימון הנדרש כדי שהפעולה תחזיר את מיקום הערך הגדול ביותר בשרשרת, החל במקום השביעי.

ב. כתבו את הזימון הנדרש כדי שהפעולה תחזיר את מיקום הערך הגדול ביותר בשרשרת myChain כולה.

תשובה :

א. בג'אווה :

```
IntNode pos = myChain;
for (int i = 0 ; i < 7; i++)
{
    pos = pos.getNext();
}
IntNode ans = getMaxPosition(pos);
```

בסישרפ :

```
IntNode pos = myChain;
for (int i = 0 ; i < 7; i++)
{
    pos = pos.GetNext();
}
IntNode ans = GetMaxPosition(pos);
```

ב. בג'אווה :

```
IntNode ans = getMaxPosition(myChain);
```

בסישרפ :

```
IntNode ans = GetMaxPosition(myChain);
```

ב – החוליה הגנרית

- רצוי לבנות את רצף ההוראה באופן שבו הגנריות תיראה טבעית ומתבקשת. בקשו מהתלמידים לממש חוליה של שלמים, אחר כך חוליה של תווים ואחר כך חוליה של מחרוזות. ערכו דיון בכיתה על כך שהשינוי קטן מאוד, ולא נוח לכתוב את המחלקה כל פעם מחדש. באופן זה, הגנריות תיראה כפתרון נוח והגיוני שילמד על הרצף, כשמרגישים את הצורך שבו (רצוי לתת לתלמידים להעלות את הצורך במנגנון כללי שכזה ורק אחר כך ללמד את הגנריות).
- השימוש בגנריות ביחידה זו הוא מוגבל ביותר ואינו מנצל את כל העוצמה של המנגנון הזה. זאת כיוון שהמטרה שלנו היא להשתמש בגנריות כדי להקל על הכתיבה של הקוד ולא כדי ללמוד את מנגנון הגנריות בפני עצמו. לכן נמנענו לדון בקשיים של מנגנון זה, ואיננו משתמשים בו כאשר אנחנו עוסקים בפעולות חיזוניות.

2.2. מחלקות עוטפות

- בסישרפ, גם טיפוסים שהם לכאורה פשוטים כמו `int` ו-`double` הם למעשה מעין עצמים, ולכן לצורך הגנריות כל טיפוס שכזה יכול לשמש טיפוס מחלקה בלי צורך בעטיפה כמו זו הנדרשת בג'אווה. בפרק מפה יתגלו החסרונות של מצב כזה, שבו לא ייתכן ש-`null` ישמש כערך של "עצם" כזה.
- סעיף זה אינו קיים בספר שבגרסת סישרפ כיוון שהנושא אינו רלוונטי לשפה. שימו לב שהסעיפים הבאים עד סעיף ג ממוספרים לפי גרסת ג'אווה.

4.4. על שימוש במחלקה הגנרית בפעולות חיצוניות

- ניתן להגדיר פעולות חיצוניות גנריות, אך התחביר הדרוש לצורך העניין מורכב ואנו לא נשתמש באפשרות זו כדי לא לסבך את התלמידים שלא לצורך. הגדרת פעולות חיצוניות כגנריות לא תפתור את הצורך שלנו בביצוע פעולות ייחודיות לערכים של `T` (כפי שמפורט בסעיף הבא) ולכן אנו מקבלים את ההחלטה של שימוש בפעולות חיצוניות קונקרטיות בלבד.
- ניתן להמחיש את הבעייתיות שבפרמטר גנרי בעזרת הדוגמה שמופיעה בפרק. אם היה אפשר לכתוב:

בג'אווה: `public static int getChainSum(Node<T> chain)`

בסישרפ: `public static int GetChainSum(Node<T> chain)`

פירוש הדבר היה שאנו יכולים לסכום שרשרת חוליות שערכה הוא טיפוס כלשהו. אך יש טיפוסים שאינם ניתנים לסכימה (כמו צב ונקודה), ועבורם הפעולה הזו חסרת משמעות. הטיפול בבעיה זו מתבצע דרך הגדרת ממשקים, `Interfaces`. ממשק מגדיר את הדרישות מהטיפוס (למשל במקרה הנדון: ניתן לסכימה). ואולם, ממשקים אינם נכללים בתוכנית הלימודים של היחידה (הם חלק מהיחידה החמישית בתכנות מונחה עצמים). זו אחת הסיבות שבגללן הוחלט לא להשתמש בטיפוסים גנריים בפעולות חיצוניות.

5.5. מימוש המחלקה הגנרית

כדאי מאוד לתת לתלמידים לממש את המחלקה. יש לשים לב שהשם המופיע בכותרת הפעולה הבונה אינו מכיל את `T`. ניתן להסביר זאת כך: `Node` הוא שם המחלקה ו-`T` הוא פרמטר של המחלקה. כיוון ששם הפעולה הבונה זהה לשם המחלקה הוא נשאר `Node` ואינו כולל את הפרמטר. נסביר בהרחבה: ניתן לראות במחלקה גנרית, לא מחלקה ממש, אלא פונקציה שמקבלת טיפוס כפרמטר ומחזירה מחלקה. כך, `Node<T>` היא פונקציה שכאשר מפעילים אותה על טיפוס קונקרטי, כגון `Integer / int`, מקבלים מחלקה קונקרטית של חוליה. ההגדרה של המחלקה הגנרית היא אם כן הגדרה של פונקציה מסוג מיוחד, ו-`T` הוא הפרמטר שלה. הסוגריים סביב הפרמטר משולשים כדי להבדיל בין פונקציה מיוחדת זו לפעולה רגילה עם פרמטרים רגילים. בכל שימוש במחלקה הגנרית מפעילים את הפונקציה על ארגומנט, שהוא טיפוס קונקרטי, ומקבלים מחלקה רגילה. כך,

Node<Integer> / Node<int> הוא ביטוי הפעלה של המחלקה הגנרית על טיפוס, והוא מייצג מחלקה רגילה. לפי השקפה זו, שמה של כל מחלקה המתקבלת מהמחלקה הגנרית הוא Node. יש להעיר כי אנלוגיה זו אמנם סבירה, ומספקת אינטואיציה מעניינת על מחלקות גנריות, אך יש לה מגבלות. בכל מקרה, היא מספקת תשובה סבירה לשאלה שאנו דנים בה כאן.

בתיעוד של מחלקות גנריות בתחילת המסמכים, לצד שם המחלקה, ניתן לזהות בבירור בשתי השפות ש-T מסומן כפרמטר של המחלקה.

ג – שרשרת חוליות כמבנה רקורסיבי

אחת ההנחות הסמויות המלוות את היחידה היא שפרמטר המועבר לפעולה אינו שווה null, אלא אם כן צוין במפורש אחרת. להנחה זו יש חשיבות כאשר באים לקבוע את תנאי העצירה של הפעולה הרקורסיבית.

נסתכל בפעולה של סכום שרשרת:

בג'אוה:

```
public static int getChainSum(Node<Integer> chain)
{
    if (chain.getNext() == null)
        return chain.getInfo();
    return chain.getInfo() + getChainSum(chain.getNext());
}
```

בסיירפ:

```
public static int GetChainSum(Node<int> chain)
{
    if (chain.GetNext() == null)
        return chain.GetInfo();
    return chain.GetInfo() + GetChainSum(chain.GetNext());
}
```

הגדרת תנאי העצירה שבחרנו מתיישבת היטב עם ההנחה הסמויה שפרמטר לפעולות חיצוניות אינו null. לכן אנו מתחילים לבדוק את ה-next של החוליה הראשונה, שקיימת בוודאות.

שימו לב שמימוש הפעולה יכול להיראות גם כך:

בג'אוה:

```
public static int getChainSum(Node<Integer> chain)
{
    if (chain == null)
        return 0;
    return chain.getInfo() + getChainSum(chain.getNext());
}
```

בסיסית:

```
public static int GetChainSum(Node<int> chain)
{
    if (chain == null)
        return 0;
    return chain.GetInfo() + GetChainSum(chain.GetNext());
}
```

גם אם תנאי זה אינו יכול להתקיים על פי הנחות היחידה בפעם הראשונה של זימון הפעולה, הרי הוא משמש לייצוג תנאי העצירה בסוף הפעולה, ולכן ניתן לכתוב כך את מימוש הפעולה.

ד – שימוש בשרשרת חוליות לייצוג של אוסף

סעיף זה חשוב מאוד ומהווה מבוא לפרק הבא. בסעיף זה אנו מייצגים אוסף ספציפי באמצעות שרשרת חוליות, ובפרק הבא נממש בעזרתה אוספים כלליים, מחסנית ותור. חשוב שהתלמידים יבינו כיצד ממומש אוסף ויתנסו בכך. כמו כן חשוב שיבינו מהו היחס בין הייצוג של האוסף לאוסף עצמו, כיצד מממשים ממשק וכיצד משתמשים בו.

פעולת ההוספה:

ייתכן שהתלמידים ירצו להוסיף תלמיד חדש לסוף הרשימה (כיוון שמי שבא מאוחר מצטרף לסוף). כדאי לדון בכך בכיתה, ולהראות שהיעילות משתפרת באופן משמעותי אם מוסיפים את התלמיד לתחילת הרשימה.

דיון:

הדיון עוסק ביתרונות של שימוש בשרשרת חוליות כייצוג של אוסף, לעומת שימוש עצמאי וישיר בשרשרת חוליות.

ה – מבנים אחרים מבוססי חוליות

סעיף זה עוסק בחוליה דו-כיוונית ובשני שימושים שלה: רשימה דו-כיוונית ועץ. ניתן להפנות את תשומת לב התלמידים לכך ששתי החוליות המרכיבות את שני המבנים האלה זהות לחלוטין (פרט לשמות התכונות: פעם אחת נקראות left ו-right ופעם אחת next ו-prev על פי ההקשר והשימוש שאנו עושים בחוליה).

חוליה דו-כיוונית ומבנים היררכיים נותנים מענה למגבלה השלישית של המערך שהעלינו בתחילת הפרק: מגבלת המבנה הסדרתי.

5. תרגילים

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

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

שאלה 1

בניית שרשרת חוליות פשוטות.
מומלץ. ניתן לעשות את הסעיפים א–ב בכיתה ולתת את סעיף ג כשיעורי בית.

שאלה 2

מגבלות בבניית שרשרת חוליות.
מומלץ. אפשר לתת כשיעורי בית.

שאלה 3

פעולה בונה מעתיקה ל-IntNode. הבנת הבעייתיות שבהוספת פעולה בונה מעתיקה למחלקה גנרית.
מומלץ.
כדאי מאוד לפתח דיון בכיתה לגבי סעיף ב. פעולה בונה מעתיקה אפשרית בתנאים שפורטו במדריך למורה לפרק 6 סעיף ז.

שאלה 4

יצירת שרשרת חוליות בתוך פעולה.
רשות. ניתן לתת כשיעורי בית.
מימוש הפעולה יאפשר לתלמידים לבדוק תוכניות אחרות העוסקות בשרשרת חוליות באופן מעשי על שרשרת קיימת.

שאלה 5

מעקב אחר קוד : הבנת פעולה הפועלת על שתי שרשרת החוליות.
רשות.

שאלה 6

מעקב אחר קוד : הבנת פעולה רקורסיבית.
מומלץ.

שאלה 7

מעקב אחר קוד : הבנת פעולה הפועלת על שרשרת חוליות.
מומלץ.

שאלה 8

כתיבת פעולה הפועלת על שרשרת חוליות.
מומלץ.

שאלה 9

מעקב אחר פעולה רקורסיבית (רקורסיית זנב) וכתיבת הפעולה באופן לא רקורסיבי.
מומלץ. את כתיבת הפעולה באופן לא רקורסיבי ניתן לתת כשיעורי בית.

שאלה 10

מעקב אחר קוד : הבנת פעולה הפועלת על שרשרת חוליות.
מומלץ.

שאלה 11

כתיבת פעולה : מיזוג של שרשרות במקום (in place), ושלא במקום (מושגים אלה יילמדו בפרק
רשימה).
רשות.

שאלה 12

איך מזהים נקודת חיבור של שרשרות של חוליות. כתיבת פעולות.
מומלץ.

שאלה 13

חזרה לתרגיל ספר הטלפונים שנעשה בפרק הקודם. הפעם ייצוג ספר הטלפונים ייעשה בעזרת שרשרת
חוליות (אוסף קונקרטי).
מומלץ.