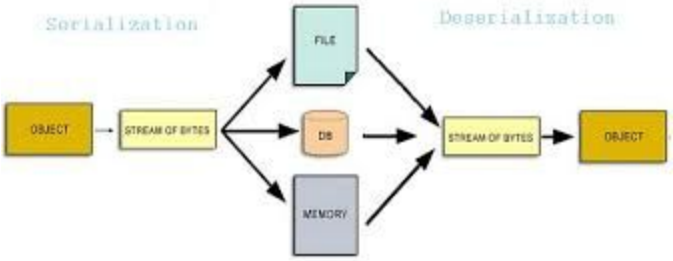
שפות תכנות 234319

אביב ה'תשע"ו



תרגיל בית מס' 5

- תאריך פרסום: 2016-05-19
- מועד אחרון להגשה: 2016-06-06
- מועד החזרה: 2016-06-16
- מתרגל אחראי: מתן פלד
- אי-מייל: mip@cs.technion.ac.il
- בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PLS2016 EX5" (ללא המרכאות).

תרגיל בית זה מורכב משני חלקים, חלק יבש וחלק רטוב. לפני ההגשה, ודאו שההגשה שלכם תואמת את הנחיות ההגשה בסוף המסמך.

נושאים

- זמן חיים של משתנים
- שפת ג'ווה
- מומלץ לקרוא על הפקודות הבאות באמצעות man, אך לא בטוח שצריך להשתמש בכולן:

time, date, factor, sed, ls, bc, dc

- מומלץ לקרוא על הנושאים הבאים ב BASH, אך לא בטוח שצריך להשתמש בכולם:

Redirection, globbing, backquotes, for loops, variables, $$, quoting

הנחיות

לבד ממסמך ההנחיות הכללי יש הנחיות מיוחדות לתרגיל זה:

- ניתן להשתמש בכל מה שלמדתם בשיעור על ML.
- רשימת הקבצים שצריכים להופיע בתוך קובץ ה-zip היא:
  `dry.pdf, hw5.sml, hw5_examples.sml`
- את התוכניות bash וכו' יש לכתוב בחלק היבש.
- שם קובץ ההגשה יהיה `EX5_ID1_ID2.zip` כאשר ID1, ID2 הם מספרי ת.ז. של המגישים בסדר עולה ממש. אם לא ניתן לסדר את מספרי ת.ז. בסדר עולה ממש יש לפנות למשרד הפנים.

חלק יבש

1. היכן יש שימוש בטיפוס None בתוך אלג' הטעינה של ה JVM? הסבירו.

קצת חזרה על החומר שנלמד בשיעור, וקצת חיפוש בויקיפדיה ובגוגל.

2. מהי שיטת ניהול הזכרון בשפת נים? האם יש איסוף אשפה? אריתמטיקה של מצביעים? משתני מחסנית הגוועים מעצמם? מה קורה בחריגה מגבולות מערך?

קצת חזרה על החומר שנלמד בשיעור, וקצת חיפוש בויקיפדיה ובגוגל. עליכם ל"זהות" את המונחים "אריתמטיקה של מצביעים", ו"משתני מחסנית "גוועים" מעצמם. לא סיפור גדול.

3. לימדו את שפת התכנות BASH:
   - וסווגו את מערכת הטיפוסים שבה לפי הקריטריונים שנלמדו בשיעור. חזרו גם על השאלה הקודמת (מס 2) ביחס לשפת BASH.
   - כיצד נקבעת נקודת התחילה של התכנית בשפת BASH? מהם "גבולות" התכנית? האם השפה אוטרקית?

תרגול זריז ומהיר של לימוד שפה שאתם כבר מכירים, שימוש במונחים המוכרים.

4. כתבו 8 תכניות קצרצרות ב BASH וב NIM המבצעות את השגיאות הבאות:
   - חלוקה ב-0.
   - גישה למקום לא חוקי במערך
   - פניה למשתנה שערכו לא מוגדר
   - גישה למשתנה רגיל כאילו הוא מערך.

סכמו את הניסיונות שלכם בטבלה שבה העמודות הבאות: סוג שגיאה, תכנית ב BASH, תכנית בנים, זמן איתור שגיאה (הרצה? קומפילציה? אחר?) והודעת השגיאה ב BASH, זמן איתור שגיאה והודעת שגיאה בנים.

הבנה מלאה יותר של שגיאות שהן "Pseudo Type Error". השוואת DYNAMIC TYPING מול STATIC TYPING. בעיקר לקרוא מחדש על המושגים, ולהפעיל אותם על שפות שאתם מכירים.

5. כתבו תכנית בשפת BASH המחשבת את מספר המונית. התכנית תדפיס את מספר הלולאות שביצעה, ואת מספר התנאים שנבדקו. מותר, אך אין חובה לתרגם מפסקל ל BASH.

תרגום של תכנית מוכרת וידועה לשפת BASH. קצת תרגול של מחזור ההרצה בשפת BASH. ענין של יותר מחצי שעה.

6. כתבו וירוס בשפת BASH. שם הקובץ יהיה virus. הרצת התכנית תהיה בלולאה. בכל איטרציה תגרום להעתקה של התכנית עצמה לקובץ בעל שם <u>יחודי</u> <u>לכל</u> <u>הרצה</u> בתיקיה tmp תמתין 5 שניות, תפעיל אותו, ומבלי להמתין לתוצאה, תמתין 5 שניות על ריק, ותחזור על הפקודות האלו בלולאה. שימו לב שגם מופעים של ההעתק צריכים לעשות משהו דומה, והקובץ בעל השם היחודי צריך להיות יחודי לכל העתק של הוירוס ולכל איטרציה שלו.

חזרה על תכנית שפועלת על עצמה (כפי שראינו בהוכחת משפט העצירה). שימוש בעקרון לכתיבת משהו משעשע. צפוי שאם לא תזהרו, תקריסו את המחשב. על כן כדאי להזהר.

7. כתבו תכנית בשפת BASH אשר תשמר בקובץ בשם measure המבצעת את הארגומנט שלה 30 פעם, ומדפיסה את זמן הריצה הממוצע (USER TIME) ואת סטיית התקן. למשל:

```
% measure ls -R
```

יבצע 30 פעם את הפקודה `ls -R`, יחשב את זמן הריצה הממוצע, ואת סטיית התקן המדגמית של זמן הריצה של הזמן. אם אחד או יותר מהביצועים נכשל, אין להתחשב בערכו. הפלט יראה כך:

```
Command: ls -R
Times: 30
Failures: 0
Average time: 1.23 sec
SD time: 0.24 sec
```

אין להדפיס יותר משלוש ספרות משמעותיות.

- הפעילו את התכנית על תכנית מספר המונית. מהם הערכים שקיבלתם.
- הפעילו את התכנית על עצמה. מהם הערכים שקיבלתם.
- הפעילו את התכנית על עצמה המופעלת על עצמה. מה הערכים שקיבלתם:

```
% measure measure measure
```

עוד חזרה על תכנית שמפעילה את עצמה על עצמה (כפי שראינו בהוכחת משפט העצירה). שימוש בעקרון לכתיבת משהו מועיל. לימוד כיצד אפשר לעשות דברים מהר, ובלי להסתבך, תוך שימוש בשפה גבוהה יותר. אולי הרכבת פקודות ב BASH. המון חיפוש בגוגל. מותר לשאול ב STACK OVERFLOW, אבל בבקשה לא לשאול איך פותרים את כל התרגיל, אלא רק נקודות קשות.

סריאליזציה בג'אווה

תרגיל זה הוא תרגיל תיאורטי/מעשי. יש בו סדרת הוראות מעשיות,שצריך לעקוב עליהן. אתם צריכים לעקוב אחרי סדרת ההוראות ולנסות כל הזמן להבין "מה הולך כאן".

התרגיל מיועד להוביל אתכם, צעד אחר צעד, לחשיפת התקשורת בין תכניות בג'ווה באמצעות סריאליזציה ותוך שמירה על שקילות שמית. זה נושא די עמוק. וגם הטריק של UUID הוא טריק חביב שכדאי להכיר אותו.

לפני ביצוע התרגיל, חשוב לקרוא שוב את הפרק על שקילות שמית לעומת שקילות מבנית. בלי זה, יהיה קשה להבין על מה ממדובר. לחזור על הפרק וההרצאה בג'ווה בזריזות.

אפשר להשתמש בלינוקס במחשב האישי, או לעבוד על t2. הערכת זמן של שעתיים שלוש לביצוע התרגיל, כולל הקריאה, על ידי שניים שעובדים יחד, שמבינים פחות או יותר את החומר התיאורטי.

בתרגיל זה נחקור את ביצוע הסריאליזציה של אובייקטים בג'ווה, ונכיר טריק שבו משתמשים, לא רק בשפה זו, כדי להתגבר על הבעיה של שקילות שמית ומבנית בתקשורת בין תכניות. תוכלו ללמוד עוד על סריאליזציה בג'ווה כאן. בכדי להבטיח תאימות, עליכם לבצע את ההנחיות כאן על מערכת לינוקס או יוניקס. שימו לב שבלינוקס ניתן לבצע העתק-הדבק מקוצר: סמן את הפקודות שאותן אתה רוצה להעתיק, עבור לחלון של מסוף, וקליק אמצעי ידביק את הפקודות. ידוע שהשתרבבה טעות אחת לתוך ההנחיות. יתכן שיותר. כדי להתמודד, עליכם להבין את משמעות כל צעד, ולא סתם לבצע את ההוראות באופן מכני. יש להצביע על הטעות/טעויות בתוך הפתרון שלכם. כדי לבצע את התרגיל, מומלץ לחזור על:

- הרצאה מס' 8: כולל סיכום על ג'ווה בדרופבוקס.
- הרצאה מס' 5: כולל סיכומים על שקילות שמית ומבנית. רצוי לקרוא שניים שלושה סיכומים. הנושא אינו קל להבנה.

בתרגיל יהיה עליך להשתמש בקבצים הבאים (שמות הקבצים אינם מופיעים כאן, אך אחרי קריאת הרצאה 8, והסיכום על הידור והרצה, יהיה קל לדעת מהם.

```
import java.io.IOException;

public class Consumer {
  public static void main(String[] args) throws IOException, ClassNotFoundException {
    if (!Common.file.exists()) {
      System.out.println(Common.now() + ") File " + Common.file + " does not exist!");
      return;
    }
    System.out.println(Common.now() + ") File " + Common.file + " exists!");
    System.out.println("I will now try to restore it:");
    Z z = Z.restore(Common.file);
    System.out.println(Common.now() + ") This is what I read\n" + z);
  }
}
```

```java
import java.io.File;
import java.util.Date;

public enum Common {
  ;
  public static final File file = new File("../delme.now");

  public static String now() {
    return new Date().toString();
  }
}
```

```java
import java.io.IOException;

public enum Producer {
  ;
  public static void main(String[] args) throws IOException, ClassNotFoundException {
    if (Common.file.exists()) {
      System.err.println(Common.now() + ") File " + Common.file + " exists!");
      System.err.println("I will now try to remove it:");
      if (!Common.file.delete()) {
        System.err.println("Cannot remove file" + Common.file);
        return;
      } else
        System.err.println(Common.now() + ") File " + Common.file + " does not exist.");
    }
    System.err.println(Common.now() + ") I will now create a new object:");
    Z z = new Z();
    System.err.println(Common.now() + ") This is what I created\n" + z);
    System.err.println(Common.now() + ") Storing created object into '" + Common.file + "'");
    z.store(Common.file);
    System.err.println(Common.now() + ") Stored!");
    System.err.println(Common.now() + ") Rereading file, I found\n" + z);
  }
}
```

```java
import java.io.*;

public class Z implements Serializable {
  private static final long serialVersionUID = -5324893836944181315L;
  // Instance fields:
```

```java
  private final String question;
  private final int answer;
  private final String cwd;
  private final String when;

  @Override public String toString() {
    return "Q: " + question + "?\n" //
        + "A: " + answer + "!\n" //
        + "(" + cwd + ", " + when + ")";
  }

  public Z() {
    cwd = System.getProperties().getProperty("user.dir");
    when = Common.now();
    question = "Life, universe, everthing";
    answer = 42;
  }

  void store(File f) throws IOException {
    store(new FileOutputStream(f));
  }

  public void store(FileOutputStream fos) throws IOException {
    store(new ObjectOutputStream(fos));
  }

  public void store(ObjectOutputStream oos) throws IOException {
    oos.writeObject(this);
    oos.close();
  }

  public static Z restore(File f) throws FileNotFoundException, IOException, ClassNotFoundException {
    return restore(new FileInputStream(f));
  }

  public static Z restore(FileInputStream fos) throws IOException, ClassNotFoundException {
    return restore(new ObjectInputStream(fos));
  }

  public static Z restore(ObjectInputStream ois) throws IOException, ClassNotFoundException {
    Z $ = (Z) ois.readObject();
    ois.close();
    return $;
  }
}
```

חלק ראשון: פרוייקט פשוט, ובו שתי תכניות היכולות להתקשר זו עם זו באמצעות סריאליזציה

1. הקם לך תיקיה חדשה לצורך ביצוע התרגיל, למשל תיקיה בשם EX5

```
rmdir -f EX5;mkdir -p EX5
```

ועבור אל התיקיה הזו

```
cd EX5
```

2. הקם תיקיה חדשה בשם Ancestor בתוך התיקיה שיצרת.

```
rm -Rf Ancestor; mkdir -p Ancestor
```

אל תעבור לתוך התיקיה Ancestor

3. עבור על ארבעת הקבצים המצורפים תוך שאתה מוודא שאתה מבין את כל מה שכתוב בהם.שים לב שניתן ליצור מהקבצים שתי תכניות.

4. צור בתוך התיקיה Ancestor עותק של הקבצים הללו. וודא שאתה בתוך התיקיה המכילה ולא בתוך התיקיה Ancestor .

5. בצע את הפקודות הללו, אשר עוברות לתיקיה Ancestor, מוחקות את תוצאות ההידור הקודם, מהדרות את הקבצים, מריצות את התכנית Producer, וחוזרות לתיקיה המכילה (EX5)

```
cd Ancestor;rm -f *.class;javac *.java;java Producer;cd ..
```

6. בדוק בפלט של הביצוע שההרצה באמת יצרה אובייקט חדש, שמרה אותו לדיסק, וקראה אותו שוב.

7. בדוק מה גודלו בבתים של הקובץ delme.now? היכן הוא מצוי? הסבר את הפלט של הפקודה

```
strings delme.now
```

8. בצע את שרשרת הפקודות הבאות, והסבר את הפלט המתקבל.

```
cd Ancestor;rm -f *.class;javac *.java;java Consumer;cd ..
```

9. כעת, בצע שיבוש מכוון בתוכן הקובץ delme.now

```
rm -f delme.now
echo "Ramanujan was here"> delme.now
```

וחזור על הפקודות מסעיף 8. בחר שורה או שתיים המייצגות בצורה הטובה ביותר את הפלט המתקבל, העתק אותן, והסבר את השגיאה במילים שלך, תוך שימוש אם יש צורך בכך, במונחים שנלמדו בכיתה.

10. חזור וצור את הקובץ delme.now:

```
cd Ancestor;rm -f *.class;javac *.java;java Producer;cd ..
```

חלק שני: הפרוייקט הפשוט מתפצל, ומתוכו נולד פרוייקט חדש, אשר יכול לקרוא אובייקטים אשר נכתבו על ידי הפרוייקט המקורי.

1. צור תיקיה נוספת בשם Consumer

```
rm -Rf Consumer;mkdir -p Consumer
```

2. העתק את מקצת הקבצים מהתיקיה Ancestor לתיקיה Consumer

```
cp Ancestor/Z.java Consumer
cp Ancestor/Consumer.java Consumer
```

```
cp Ancestor/Common.java Consumer
```

3. בדוק שהתכנית שבתיקיה Consumer יכולה לקרוא את הפלט של התכנית שבתיקיה Ancestor. כיוון שנעשה זאת פעמים רבות, נגדיר קיצור לשם כך:

```
alias consume="cd Consumer;rm -f *.class;javac *.java;java Consumer;cd .."
```

וכעת נכתוב:

```
consume
```

3. השתמש בעורך הטקסט החביב עליך, כדי לשנות את ערכו של המשתנה serialVersionUID שבקובץ Z.java שבתיקיה Ancestor למספר 1729, למשל

```
gvim B/Z.java
```

4.בדוק שוב שהתכנית שבתיקיה Consumer יכולה לקרוא את הפלט של התכנית שבתיקיה Ancestor

```
consume
```

הסבר את הודעות השגיאה המתקבלות: בחר שורה או שתיים המייצגות בצורה הטובה ביותר את הפלט המתקבל, העתק אותן, והסבר את השגיאה במילים שלך, תוך שימוש אם יש צורך בכך, במונחים שנלמדו בכיתה.

חלק שלישי: הפרוייקט המקורי מוליד פרוייקט חדש, אשר יכול לכתוב אובייקטים אשר יקראו על ידי הבן האחר של הפרוייקט המקורי. הפרוייקט המקורי נמחק

1. צור תיקיה נוספת בשם Producer

```
rm -Rf Producer;mkdir -p Producer
```

2. העתק קבצים מהתיקיה Ancestor לתיקיה Producer

```
cp Ancestor/Z.java Producer
cp Ancestor/Producer.java Producer
cp Ancestor/Common.java Producer
```

3. מחק את הפרוייקט המקורי:

```
rm -Rf Ancestor
```

4.הכן קיצור לשם הרצת הפרוייקט החדש שיצרת המקורי:

```
alias produce="cd Producer;rm -f *.class;javac *.java;java Producer;cd .."
```

5. בצע את הקיצור כאמור:

```
produce
```

6. הרץ את התכנית שבתיקיה Consumer ונסה לקרוא את הקובץ שנשמר באמצעות התכנית שבתיקיה Producer

```
consume
```

7. השתמש בעורך הטקסט החביב עליך, כדי לשנות את ערכו של המשתנה serialVersionUID שבקובץ Z.java שבתיקיה C למספר 1729, למשל

```
gvim Producer/Z.java
```

ובדוק כעת שהתכנית שבתיקיה B יכולה לקרוא את הקובץ שנשמר מהתכנית שבתיקיה C

```
produce;consume
```

חלק רביעי: שני הפרוייקטים משנים את הפורמט באופן בלתי תלוי.

1. הוסף שדה מטיפוס int ובשם a למחלקה Z שב-Producer. בדוק שהתקשורת ממשיכה להתקיים:

```
produce;consume
```

2. הוסף שדה מטיפוס double ובשם x למחלקה Z שב-Consumer. בדוק שהתקשורת בין שני הפרוייקטים ממשיכה להתקיים:

```
produce;consume
```

3. שנה את שם השדה שהוספת מטיפוס double למחלקה Z שב-Consumer, לשם a (כזכור הוגדר שדה בשם a אך מטיפוס int בפרוייקט ה Producer) בדוק והסבר את הודעת השגיאה

```
produce;consume
```

4. שנה ל int את טיפוס השדה במחלקה Z שב-Consumer. בדוק שהתקשורת חוזרת להתקיים:

5. מחק את השדה serialVersionUID שבפרוייקט ה-Producer. שים לב ששני הפרוייקטים מפסיקים להתקשר:

```
produce;consume
```

כאשר השדה serialVersionUID חסר, מערכת זמן הריצה של Java מוסיפה ערך המחושב מערבול החתימה של הטיפוס (שדות, פונקציות וכו'). מצא מתוך הודעת השגיאה את הערך שמערכת זמן הריצה של Java הוסיפה עבור הטיפוס Z

6. מחק את השדה serialVersionUID שבפרוייקט Consumer. שים לב ששני הפרוייקטים חוזרים להתקשר:

```
produce;consume
```

אם יש תקלות בתקשורת,בדוק ששתי ההגדרות זהות. ואם לא, העתק את הקובץ Z.java מאחד הפרוייקטים אל האחר.

7. שנה את סדר השדות באחד העותקים של Z.java ובדוק שהתקשורת ממשיכה להתקיים.

8. הוסף פונקציה ריקה לאחד העותקים של Z.java

```
void f() {}
```

ובדוק שהתקשורת פסקה:

9. סכם מסקנותיך בפירוט ביחס לחישוב האוטומטי של ה serialVersionUID

10. סלק את הביטוי implements Serializable תחילה מהעותק שב Consumer ואחר מזה שב Producer, הרץ, וסכם מסקנותיך בהרחבה ולא בתמצות.

חלק רטוב

שאלה 1: רשימות מתחלפות

רקע: הטיפוס 'a list בשפת ML מוגדר כך:

```
datatype 'a list = nil | :: of 'a * 'a list
infixr 5 ::
```

(infixr הוא infix עם אסוציאטיביות ימנית. 5 הוא ספרה שמגדירה את העדיפות בין אופרטורים.)

פרט לכך, ML מגדירה תחביר מיוחד עבור רשימות, בעזרת סוגריים מרובעים:

```
[1,2,3] = 1::2::3::nil
```

החיסרון של רשימות ביחס ל tuple, למשל, היא שכל האיברים של רשימה נתונה הם מאותו טיפוס.

בתרגיל זה ננסה לתקן את המצב.

1. הגדירו טיפוס גנרי "רשימה מתחלפת". ערכים מטיפוס זה מייצגים רשימות המחזיקות איברים מטיפוס 'a, אחריו איבר מטיפוס 'b, אחריו איבר מטיפוס 'a וכן הלאה. רשימות עשויות להיות בכל אורך שהוא, כולל 0.

   טיפוס שיאפשר לבצע את המשימות הנדרשות בהמשך השאלה יקיים את הדרישה.

   השלימו את התבנית עבור הטיפוס והוסיפו את ההגדרה לקובץ הפתרון:

```
datatype ('a, 'b) heterolist = _______
                             | _______
```

אין לחרוג מצורה זאת - על ההגדרה להכיל רק תו | בודד. אין להשתמש בטיפוס list.

2. בשפת ML לא ניתן להגדיר תחביר מקביל העושה שימוש בסוגריים מרובעים, אך אם היה ניתן להגדיר תחביר כזה היה אפשר ליצור רשימות כגון

```
val x1y2 : (string, int) heterolist  = ["x",1,"y",2]
```

הגדירו את הפונקציה

```
build4 : 'a*'b*'a*'b -> ('a, 'b) heterolist
```

שתחזיר רשימה מתחלפת, המחזיקה בדיוק את ארבעת הערכים שהועברו בפרמטר.

הפיתרון צריך להינתן בביטוי בודד (אין להגדיר פונ' עזר פנימיות וכו'):

```
fun build4 (x,one,y,two) = _____________
```

3. (לא להגשה): השלימו את התבנית בהצהרה להלן כך שכל אחד מהמשתנים x, one, y, two יחזיק את הערך המתאים לשמו, ושתי השורות יתקמפלו וירוצו ללא חריגה (כאשר הן מבוצעות ברצף):

```
val _______________ = build4 ("x",1,"y",2)
val ("x",1,"y",2) = (x,one,y,two)
```

מומלץ מאוד לא להמשיך לסעיפים הבאים לפני שהצלחתם את החלקים הקודמים.

4. הגדירו פונקציה בחתימה להלן (במדויק):

```
unzip : ('a, 'b) heterolist  -> 'a list * 'b list
```

על כל רשימה בזוג המוחזר להחזיק בדיוק את כל האיברים מהטיפוס המתאים ברשימה שהתקבלה כפרמטר, ולפי אותו סדר. פונקציה זאת תמיד תחזיר ערך (לא ייזרקו חריגות)

בפרט, על השורה הבאה להתקמפל ולרוץ ללא שגיאות וללא הודעות מיוחדות:

```
val (["x","y"], [1,2]) = unzip (build4 ("x",1,"y",2))
```

5. הגדירו פונקציה `zip : 'a list * 'b list -> ('a, 'b) heterolist` המבצעת את הפעולה ההפוכה לפונקציה בסעיף הקודם. במקרה שמספר האיברים ברשימות איננו מתאים (אם לא מתקיים שם `len(a) - 1 ≤ len(b) ≤ len(a)`) יש לזרוק את החריגה Empty. הפונקציה `id_heterolist` להלן (שאיננה להגשה) שקולה לפונקציית הזהות על `('a, 'b) heterolist`:

```
val id_heterolist  = zip o unzip (* ignore the warning *)
```

וכן הקוד הבא צריך להתקמפל ולרוץ ללא שגיאות וללא הודעות מיוחדות:

```
val (["x","y"], [1,2]) = unzip (zip (["x", "y"], [1,2]))
```

שאלה 2: רצפים

לכל אורך השאלה, נשתמש בהגדרה הבאה של רצף (כפי שנלמד בתרגולים):

```
datatype 'a seq = Nil | Cons of 'a * (unit-> 'a seq);
```

ובהגדרות הבאות של פונקציות head/tail (שהן שונות קצת מאלו שהוצגו בתרגול):

```
exception EmptySeq;
fun head(Cons(x,_)) = x | head Nil = raise EmptySeq;
fun tail(Cons(_,xf)) = xf() | tail Nil = raise EmptySeq;
```

עליכם להוסיף את ההגדרות הללו בתחילת קובץ הפתרון.

בשאלה זו נעסוק בהרחבה של טיפוס הרצף שנלמד בכיתה, הנקראת "רצף דו-כיווני".

רצף כזה הוא רצף שניתן להתקדם בו קדימה ואחורה. לשם כך, נגדיר את שני ה- datatypes הבאים:

```
datatype direction = Back | Forward;
datatype 'a bseq =   bNil
                   | bCons of 'a * (direction -> 'a bseq);
```

כאשר bNil מייצג את הרצף הריק ו- bCons מייצג איבר ברצף, המכיל ערך ופונקציית התקדמות.

פונקציה זו מקבלת ערך מטיפוס direction (Back או Forward) ומחזירה את האיבר הקודם או הבא ברצף, בהתאמה.

נגדיר בנוסף את שלושת הפונקציות הבאות:

```
fun bHead(bCons(x,_)) = x | bHead bNil = raise EmptySeq;
fun bForward(bCons(_,xf)) = xf(Forward) | bForward bNil = raise EmptySeq;
fun bBack(bCons(_,xf)) = xf(Back) | bBack bNil = raise EmptySeq;
```

יש להוסיף את ה datatypes והפונקציות שהוגדרו לעיל לקובץ הפתרון שלכם.

1. ממשו את הפונקציה `intbseq : int -> int bseq`, המקבלת מספר שלם x ומחזירה "רצף דו-כיווני" אינסופי המייצג את כל המספרים השלמים, כאשר האיבר הנוכחי ברצף המוחזר מכיל את הערך x. לכל איבר y ברצף, האיבר העוקב לו הוא y+1 והאיבר הקודם לו הוא y-1 .

דוגמת הרצה:

```
- intbseq 2;
val it = bCons (2,fn) : int bseq
- bForward(it);
val it = bCons (3,fn) : int bseq
- bForward(it);
val it = bCons (4,fn) : int bseq
- bBack(it);
```

```
val it = bCons (3,fn) : int bseq
- bBack(it);
val it = bCons (2,fn) : int bseq
- bBack(it);
val it = bCons (1,fn) : int bseq
- bBack(it);
val it = bCons (0,fn) : int bseq
- bBack(it);
val it = bCons (~1,fn) : int bseq
```

2. ממשו את הפונקציה `bmap : ('a -> 'b) -> 'a bseq -> 'b bseq`, המקבלת פונקציה f ו"רצף דו-כיווני", ומחזירה "רצף דו-כיווני" שבו כל איבר עם ערך x הופך לאיבר עם ערך f(x).

דוגמת הרצה:

```
- bmap (fn x =>x*x) (intbseq 2);
val it = bCons (4,fn) : int bseq
- bForward(it);
val it = bCons (9,fn) : int bseq
- bForward(it);
val it = bCons (16,fn) : int bseq
- bBack(it);
val it = bCons (9,fn) : int bseq
- bBack(it);
val it = bCons (4,fn) : int bseq
- bBack(it);
val it = bCons (1,fn) : int bseq
- bBack(it);
val it = bCons (0,fn) : int bseq
- bBack(it);
val it = bCons (1,fn) : int bseq
- bBack(it);
val it = bCons (4,fn) : int bseq
```

3. ממשו את הפונקציה:

```
bfilter : ('a -> bool) -> direction -> 'a bseq -> 'a bseq
```

המקבלת פונקציית פרדיקט ,איבר d מטיפוס direction ו"רצף דו-כיווני". הפונקציה מחזירה "רצף דו-כיווני" המכיל רק איברים עם ערכים שהפרדיקט מחזיר עבורם true. אם האיבר הנוכחי ברצף

שהתקבל אינו ברצף המוחזר, אזי האיבר הנוכחי ברצף המוחזר יהיה האיבר הקרוב ביותר מכיוון d שהפרדיקט מחזיר עבורו true (כלומר, אם d=Forward, יוחזר "האיבר הבא" הקרוב אליו ביותר שהפרדיקט מחזיר עבורו true, אחרת יוחזר "האיבר הקודם" הקרוב אליו ביותר שהפרדיקט מחזיר עבורו true). אם לא נמצא איבר בכיוון d שהפרדיקט מחזיר עבורו true והרצף הוא אינסופי תכנס הפונקציה ללולאה אינסופית.

דוגמת הרצה:

```
- bfilter (fn x => x mod 2 = 0) Back (intbseq 2);
val it = bCons (2,fn) : int bseq
- bForward(it);
val it = bCons (4,fn) : int bseq
- bForward(it);
val it = bCons (6,fn) : int bseq
- bBack(it);
val it = bCons (4,fn) : int bseq
- bBack(it);
val it = bCons (2,fn) : int bseq
- bBack(it);
val it = bCons (0,fn) : int bseq
- bBack(it);
val it = bCons (~2,fn) : int bseq
- bfilter (fn x => x mod 2 = 0) Back (intbseq 1);
val it = bCons (0,fn) : int bseq
- bfilter (fn x => x mod 2 = 0) Forward (intbseq 1);
val it = bCons (2,fn) : int bseq
```

4. ממשו את הפונקציה `seq2bseq : 'a seq -> 'a seq -> 'a bseq`, המקבלת שני רצפים חד-כיווניים, כאשר לרצף הראשון נקרא הרצף ההפוך ולשני נקרא הרצף הרגיל. הפונקציה תחזיר "רצף דו-כיווני", כאשר הערך של האיבר הנוכחי ברצף המוחזר הוא הערך של האיבר הראשון ברצף הרגיל (הפרמטר השני שהתקבל) כל האיברים הבאים אחרי אותו איבר הם האיברים הבאים אחריו ברצף הרגיל. כל האיברים הקודמים לו הם האיברים המופיעים ברצף ההפוך, לפי סדר הופעתם (כלומר, האיבר שקודם לאיבר הנוכחי הוא האיבר הראשון ברצף ההפוך, האיבר שלפניו הוא האיבר השני ברצף ההפוך וכן הלאה). ניתן להניח ששני הרצפים המתקבלים הם אינסופיים, ולכן אין צורך לטפל במקרה שמתקבל Nil.

דוגמת הרצה:

נגדיר את שתי הפונקציות הבאות, המחזירות רצפים חד-כיווניים המכילים את כל המספרים השלמים בסדר עולה ויורד, בהתאמה:

```
fun from(x) = Cons(x, fn()=>from(x+1));
fun downfrom(x) = Cons(x, fn()=>downfrom(x-1));
```

כעת נגדיר את הרצף הדו-כיווני המכיל את כל המספרים השלמים, השקול לרצף שהגדרנו בסעיף א' (כאשר האיבר ההתחלתי הוא 0) באופן הבא:

```
-seq2bseq (downfrom ~1) (from 0);
val it = bCons (0,fn) : int bseq
```

כעת ניתן להקיש את רצף הפקודות הבא, ולקבל את הפלטים הרשומים:

```
-bForward(it);
val it = bCons (1,fn) : int bseq
-bBack(it);
val it = bCons (0,fn) : int bseq
-bBack(it);
val it = bCons (~1,fn) : int bseq
```

5. ממשו את הפונקציה bSeqJump : 'a bseq -> int -> 'a bseq המקבלת "רצף דו-כיווני" אינסופי ומספר שלם וחיובי m, ומחזירה "רצף דו-כיווני" המכיל רק את האיברים מרצף הקלט שהאינדקס של המיקום שלהם מתחלק ב-m ללא שארית. למשל, אם m=2 אז הרצף המוחזר יכיל רק את האיברים מהרצף המקומי הנמצאים במקומות זוגיים. ניתן להניח שהרצף המתקבל הוא אינסופי ושהמספר m הוא חיובי ממש (גדול מאפס) ואין צורך לבדוק זאת. בנוסף הניחו שהמיקום של האיבר ההתחלתי של רצף הקלט הוא 0.

```
- intbseq 0;
val it = bCons (0,fn) : int bseq
- bSeqJump it 3;
val it = bCons (0,fn) : int bseq
- bForward it;
val it = bCons (3,fn) : int bseq
- bForward it;
val it = bCons (6,fn) : int bseq
- bBack it;
val it = bCons (3,fn) : int bseq
- bBack it;
val it = bCons (0,fn) : int bseq
- bBack it;
val it = bCons (~3,fn) : int bseq
```

שאלה 3: מספר מונית עם רצפים (כן! עוד פעם! מוהאהא)

ממשו את הפונקציה taxiCabSeq : unit -> int seq המקבלת unit ומחזירה את רצף מספרי המוניות החל מ-2 (הסדרה המתוארת פה: https://oeis.org/A011541).

```
- taxiCabSeq ();
val it = Cons (2,fn) : int seq
- tail it;
val it = Cons (1729,fn) : int seq
- tail it;
val it = Cons (87539319,fn) : int seq
```

בהצלחה!