שפות תכנות, 234319

חורף תשע"ח

תרגיל בית מספר 3 - Advanced Typing, ML Lists

תאריך פרסום27.11.2017 :
מועד אחרון להגשה11.12.2017 :
מועד אחרון להגשה מאוחרת14.12.2017 :
מתרגל אחראי: תומר גלבר
אי-מייל: tomerghelber@cs.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (**subject**) יהיה "**3PLW2017 EX**" (ללא המרכאות).

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

חלק יבש

1. **עיינו שוב בשפת Kotlin.**

א. סווגו את מערכת הטיפוסים של שפת Kotlin לפי הקריטריונים שנלמדו בתרגול ולפי פרק 4 של חוברת ההרצאות.

הטיפוסים הקיימים:

- טיפוסיות חזקה או חלשה - **טיפוסיות באמצע, כלומר לא חזקה כמו Pascal אבל גם לא חלשה כמו C,** לדוגמא ניתן לחבר משתנה מסוג int למשתנה מסוג double, ואפשר לבצע המרות, אבל אי אפשר לעשות השמה של bool לתוך int.
- טיפוסיות סטטית או דינמית - **טיפוסיות סטטית**. כלומר הטיפוסיות נבדקת בזמן הקומפילציה, אבל בנוסף קיימת פקודת dynamic שמבטלת את הבדיקה ואז תיתכן האפשרות לבדיקת הטיפוסים בזמן ההרצה.
- אחריות סיווג טיפוסים – בקוטלין קיימות שתי האפשרויות, הן פירוש של הקומפיילר והן הכרזה על סוג הטיפוס. כלומר בדומה לML.
- אפליה – לא קיימת אפליה בשפה, כלומר קוטלין לא מפלה בין פונקציות לטיפוסים(בעיקר בגלל שקוטלין מוכוונת OOP, כלומר כל טיפוס בשפה הוא אובייקט).
- שקילות שמית/מבנית – בקוטלין יש גם שקילות שמית וגם שקילות מבנית.

ב. האם קיימים הטיפוסים הבאים או קירובם בשפת Kotlin? הסבירו:

- **Unit** - טיפוס מורכב - התוצאה של 0 אלמנטים.
  קיים בkotlin – טיפוס עם ערך יחיד בלבד, בדומה לטיפוס ה-void בjava.
- **None** - טיפוס מורכב - איחוד זר של 0 אלמנטים.
  קיים בkotlin – טיפוס בשם *nothing*.
- **Any** - איחוד זר של כל הטיפוסים בשפה:
  קיים בkotlin - קבוצת העולם של כל class היא any.

2. **הסבירו מהם ההבדלים בין טיפוסיות שמית לבין טיפוסיות מבנית. התייחסו בתשובתכם ליתרונות ולחסרונות של כל אחת מהגישות.**

טיפוסיות שמית nominal typing, name equivalence - שני טיפוסים שקולים רק אם יש להם את "אותו השם" והם הוגדרו ב"אותו המקום". באופן יותר מדויק - שקילות שמית בין שתי ישויות מתקיימת אם הם הוגדרו ע"י אותו טיפוס ממש.
טיפוסיות מבנית structural typing - אם המבנה הפנימי של שני טיפוסים הוא זהה – כלומר אם התחלנו מהטיפוסים האטומיים ובנינו שני טיפוסים בדיוק באותו האופן, אז גם הטיפוסים נחשבים שקולים.
ההבדל הוא שטיפוסיות שמית מתקיימת <u>אם שני הטיפוסים הוגדרו ע"י אותו הטיפוס ממש</u>, וטיפוסיות מבנית מתקיימת <u>אם המבנה הפנימי של שני טיפוסים הוא זהה</u>.

חסרונות של הטיפוסיות השמית - כאשר תוכנית המחולקת למספר קבצים, לעיתים כל החלקים צריכים להכיר טיפוס מסוים. במקרה שבו מספר קבצים צריכים להכיר טיפוס, נשאלת השאלה היכן יש להגדיר את הטיפוס. אם יוגדר באחד הקבצים, הטיפוס לא יוכר בקבצים האחרים. אם יוגדר בכולם, לא יהיה זה אותו טיפוס, כיוון שההגדרה אינה באותו מקום ולכן לפי השקילות השמית הטיפוסים שונים.

בעת כתיבה לקבצים או קריאה מהם, נקראים ערכים מטיפוסים מסוימים. בעת כתיבת ערך על ידי תוכנית אחת נכתב ערך מטיפוס אחד, ובעת קריאתו על ידי תוכנית שנייה הוא נקרא כטיפוס אחר. כיוון שהטיפוסים לא הוגדרו באותו מקום אלו טיפוסים שונים לפי השקילות השמית. גם אם זוהי אותה תוכנית המנסה לקרוא קובץ שכתבה בהרצה קודמת, נוצרת אותה בעיה כיוון שכל הרצה יוצרת טיפוס חדש, ולכן זהו לא אותו טיפוס בשתי ההרצות של התוכנית.

יתרון לטיפוסיות מבנית- אין צורך לתת שמות לטיפוסים, אלא רק לצרכי המתכנת בלבד ולכן ניתן לעשות שינויים בשם בזמן התוכנית.

3. מה ההבדל בין טיפוס לבין ערך? הגדירו כל אחד מהם והוסיפו דוגמאות.

ערך הוא ישות שקיימת במהלך חישוב וריצת התוכנית. קבוצת הערכים בשפה מוגדרת רקורסיבית. ניתן להצמיד ערך למשתנה.
טיפוס הוא קבוצה של ערכים. כל טיפוס הוא תת קבוצה של קבוצת עולם הערכים של השפה, מכך נובע כי הטיפוסים מוגדרים רקורסיבית, כאשר יש לנו בשפה טיפוסים אטומיים וכן בנאי טיפוסים. כל טיפוס מגדיר קבוצת ערכים וחוקיות, כלומר, אילו ערכים יכול משתנה לקבל, ומהי קבוצת הפעולות אותן ניתן להפעיל על הטיפוס.
כל טיפוס T מגדיר את S(T), קבוצת הערכים T.
השונה בין ערך לטיפוס הוא שטיפוס מגדיר את קבוצת הערכים אותם משתנה מטיפוס זה יכול לקבל, ומהי קבוצת הפעולות אותן ניתן להפעיל על משתנים מסוג טיפוס זה. אך קבוצה של ערכים אינה יכולה להגדיר טיפוס.

דוגמאות:
שפת C

```
int a = 5;
```

במקרה זה, int הוא הטיפוס והערך הוא 5. int הוא טיפוס שנועד לאכסן מספרים שלמים. 5 שייך לקבוצת הערכים השלמים, ולכן המשתנה a יכול לקבל את הערך 5.

```
char a = 'a';
```

בדוגמא זו, char הוא הטיפוס והערך הוא a. char הוא הטיפוס הבסיסי לייצוג תווים. a שייך לקבוצת התווים, ולכן המשתנה a יכולה לקבל את הערך a.

האם טיפוס יכול להיות ערך? לאילו בנאים תאורטיים הם מתאימים?
לדוגמא, ב++C טיפוס יכול להיות ערך. הפעולה typeld מחזיר לאת סוג הטיפוס של המשתנה,
כלומר עבור int a, יהיה typeld(a) = int.
בדומה גם בקוטלין הפעולה is מבצעת פעולה זהה.

רשימת בנאים תאורטיים:
מיפוי, איחוד זר, record, P(A), מכפלה קרטזית

4. **הסבירו מה ההבדל בין type aliasing לבין type branding, ותנו דוגמה לתכנית שבה ההבדל הזה חשוב (לא חייבים לתת דוגמת קוד).**

Type aliasing - מתן שם נוסף לטיפוס קיים. לאחר מתן השם ניתן להשתמש השני השמות של הטיפוס. קיימת שקילות מבנית בין הטיפוס וה-alias שלו וניתן לבצע עליהם את אותן הפעולות, כאילו היו אותו הטיפוס בדיוק.
Type branding - יצירת טיפוס חדש על סמך טיפוס אחר קיים. Type branding משתמש בשקילות שמית ולכן גם אם ניצור שני טיפוסים זהים מבחינה מבנית, הם לא יחשבו זהים מבחינת השפה.
ההבדל בין type aliasing לבין type branding מושרש בכך ש type aliasing משתמש בשקילות מבנית ו- type branding משתמש בשקילות שמית. שני טיפוסים Type aliasing יחשבו זהים מבחינת השפה, לאומת שני טיפוסים Type branding שלא ייחשבו זהים מבחינת השפה.

אם נגדיר בשפת פסקל שני טיפוסים int חדשים באמצעות TYPE, לכאורה אפשר לחשוב שניתן לבצע בניהם פעולות כמו חיבור וכפל, אך זה לא נכון מכיוון שtype משתמש בbranding.

5. **הגדירו מהי שגיאת טיפוס.**

שגיאת טיפוס היא ניסיון של התוכנית לבצע על ערך מסוג טיפוס מסויים:
- שינוי של הערך בצורה שלא עקבית עם סוג הטיפוס.
- פירוש של הייצוג של המכונה של הערך בצורה שלא עקבית עם הטיפוס.

ישנן שתי צורות לדיווח על טיפוס שגיאה:
- דינאמי – בזמן ריצה, ("זיהוי שגיאה ממש לפני שהיא קורת/ בזמן שהיא קורת") כלומר התוכנית מבצעת לפני כל צעד בריצה בדיקה של הפעולה והערכים.
  <u>לדוגמא השפות:</u> ruby, perl, python, smalltalk
- סטטית – בזמן הידור, מניעת השגיאה, ע"י פסילה מראש של תוכניות העלולות לבצע שגיאת טיפוס אם יורשו לרוץ. שפות עם שגיאת טיפוס סטטית דורשות התחייבות לטיפוס, כלומר, הצהרה מראש על סוג המשתנה.
- <u>לדוגמא השפות:</u> pascal, c, c++, java, C#

אם קוד מסוים מבצע שגיאת טיפוס, האם בהכרח הקוד לא ירוץ? תנו דוגמאות קוד קצרות לכל אחד מהמקרים האפשריים משפות לבחירתכם.

דוגמא לשגיאה בזמן קומפילציה:

```
Int a = false+13;
```

דוגמא לשגיאה בזמן ריצה:

```
double a = 0;
double b = 10/a;
```

חלק רטוב

חלק 2.1 - ML Lists - חימום

בחלק הראשון נממש פונקציות ואופרטורים, ניתן להיעזר בפונקציות מסעיפים קודמים.
כל המטריצות בסעיפים בחלק זה יהיו גדולות בגודלן מ-0. כל הרשימות יהיו מטריצות תקינות.

1. ממש פונקציה בשם T המקבלת מטריצה ועושה לה שרבוב (transpose).

val T = fn : (real list list) -> real list list

2. ממש פונקציה בשם invertible המקבלת מטריצה (הקלט תקין - המטריצה הפיכה) ועושה לה היפוך.

val invertible = fn : (real list list) -> real list list

3. ממש את האופרטור ++ אשר מחבר שתי מטריצות. הניחו כי גודלן נכון.

val ++ = fn : (real list list) * (real list list) -> (real list list)

4. ממשו את האופרטור ** אשר מכפיל את מטריצות (כפל מטריצות ולא איבר איבר). הניחו כי הקלט נכון.

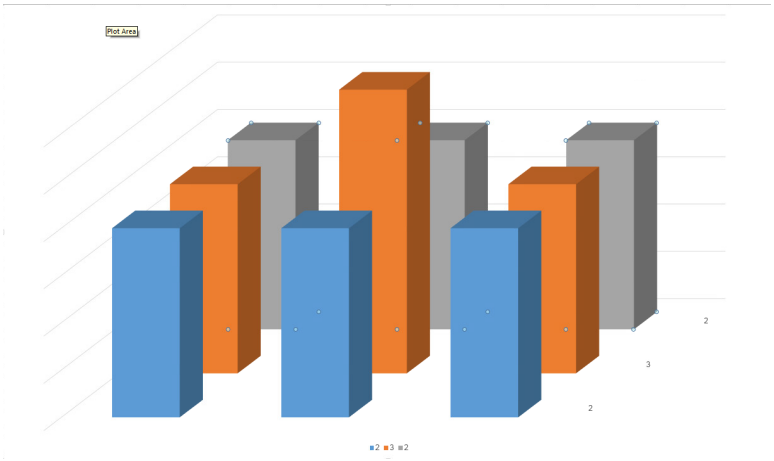
val ** = fn : (real list list) * (real list list) -> (real list list)

חלק 2.2 - ספר מה ראית, ואבנה לך עיר

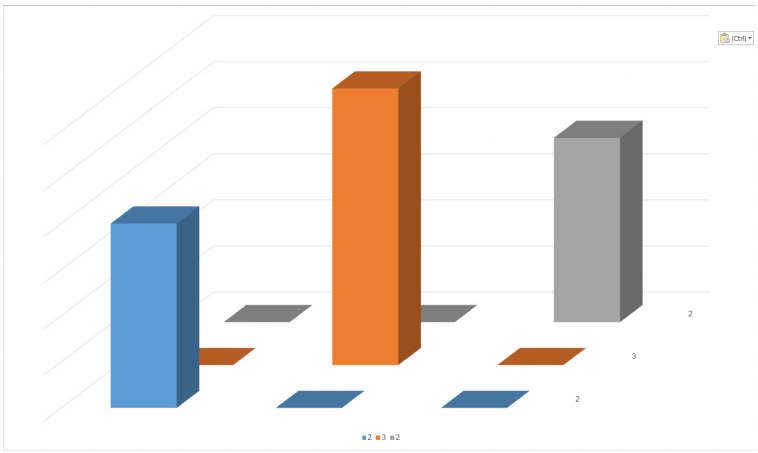
בתרגיל זה ננסה להבין מה גובה בנייני העיר, אשר בנויה כשתי וערב, ע"י תיאור של חבר. התיאור שלו כולל 2 מערכים המציינים את תמונת הנוף של עיר - הראשון מצפון והשני ממערב. בכל תא יש מספר המתאר את מספר הקומות המקסימלי שנצפה בעבור אותה שדרה מהכיוון המתאים (צפון/מערב).
המספרים אי שליליים (גדולים או שווים ל-0).

בגלל שהחבר ראה רק שני צדדים, נוכל לדמיין לעצמנו מגוון גדול מאוד של ערים שונות מאותו תיאור.
בשאלה שלנו נסתכל על הערים הקיצוניות - אחת מקסימלית (מקסימום קומות) ואחת מינימלית (עם מינימום קומות).

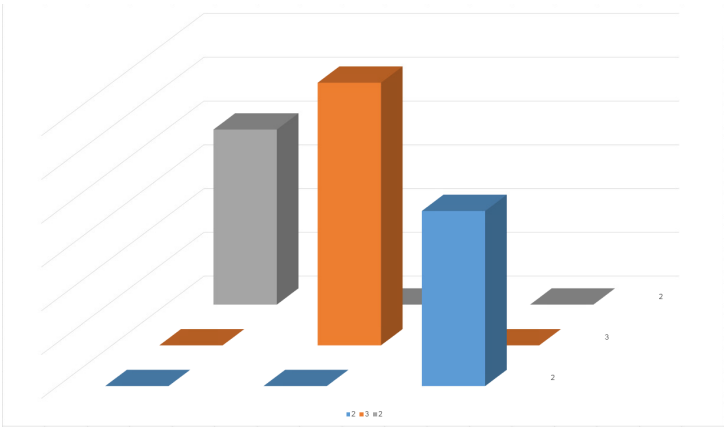
הנה דוגמה קיצונית מקסימלית עבור [2,3,2], [2,3,2] - רק במקום אחד יכול להיות בניין בן 3 קומות, וכל השאר יהיו מקסימום המתאפשר מההגדרה והם בעלי 2 קומות:



הנה דוגמה קיצונית מינימלית עבור אותו הקלט אבל - רק במקום אחד יכול להיות בניין בן 3 קומות, אבל מספיקים לנו רק שני בניינים בני 2 קומות כדי לענות על ההגדרה:



שימו לב שיש עוד קיצונית מינימלית (שקולה):



שתיהן קבילות, וצריך להחזיר רק אחת.

הפונקציות שתייצרו יראו כך:

max_city west north

max_city = fn (int list) -> (int list) -> (int list list)

min_city west north

min_city = fn (int list) -> (int list) -> (int list list)

	N	O	R	T	H
W					
E					
S					
T					

דוגמאות הרצה:

max_city [1,2,3,4] [1,2,3,4]
[[1,1,1,1],[1,2,2,2],[1,2,3,3],[1,2,3,4]]
min_city [1,2,3,4] [1,2,3,4]
[[1,0,0,0],[0,2,0,0],[0,0,3,0],[0,0,0,4]]

max_city [1,2,1] [1,2,1]
[[1,1,1],[1,2,1],[1,1,1]]
min_city [1,2,1] [1,2,1]
[[1,0,0],[0,2,0],[0,0,1]]

max_city [1,2,4,4] [1,2,3,4]
[[1,1,1,1],[1,2,2,2],[1,2,3,4],[1,2,3,4]]
min_city [1,2,4,4] [1,2,3,4]
[[1,0,0,0],[0,2,0,0],[0,0,3,4],[0,0,0,4]]
This is an alternative output - [[1,0,0,0],[0,2,0,0],[0,0,0,4],[0,0,3,4]]

max_city [1,2,4] [1,2,3,4]
[[1,1,1,1],[1,2,2,2],[1,2,3,4]]
min_city [1,2,4] [1,2,3,4]
[[1,0,0,0],[0,2,0,0],[0,0,3,4]]

חלק (ממש) רטוב

הורידו את הקובץ וחלצו את הקובץ REPY. הקובץ מכיל את כל הקורסים של הסמסטר והשעות שתוכנות כמו ChoiceFreak, ttime ו-UDonkey משתמשות בו כדי לדעת מתי ההרצאות והתרגולים.
בתרגיל עליכם להבין מה הקידוד של קובץ ה-REPY, ולהמיר לקידוד חדש - ISO-8859-8.
את ההמרה תכתבו בשפה Kotlin ע"י מילוי הפונקציה completeMe, אשר מקבלת מערך של בתים ומחזירה מערך של בתים.

```kotlin
import java.io.File

fun main(args: Array<String>) {
   // Change this to read the right file
   val inputPath = "/home/tomer/work/repy/src/REPY"
   val outputPath = "/home/tomer/work/repy/src/output"
   File(outputPath).writeBytes(completeMe(File(inputPath).readBytes()))
}

fun completeMe(input: ByteArray): ByteArray {
   // TODO: Complete the function
   val output:ByteArray = input
   return output
}
```

את הקוד שלכם יש להגיש בקובץ בשם ex3.kt. רק הפונקציה comleteMe תבדק, תחת קימפול ל-JVM ואין להשתמש בחבילות חיצוניות.

בונוס (ארוך, קשה ומיותר — לא חובה להגיש)

כתבו חבילה שתפרסר את הפורמט של הקובץ ותייצר אובייקטים המכילים את המידע בצורה נגישה. יש לכם יד חופשית לתכנן ולממש את החבילה כיד רוחכם, רק נדרוש שתכללו קובץ README.md בפורמט markdown שמסביר את ההגשה שלכם. זכרו שאנחנו יכולים לתת ניקוד רק לדברים שאנחנו מבינים. את החבילה שימו בספריה חדשה בשם lib_bonus. יינתן בונוס לכל מי שמגיש חבילה עובדת, ובונוס משמעותי לזוג שיגיש חבילה אלגנטית.

הנחיות הגשה

- בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד (כולל) תרגול 6. אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים..
- שימו לב כי ML היא case-sensitive, ודאו על כן כי שמות הפונקציות הן בדיוק כפי שנדרש בתרגיל.
- ההגשה האלקטרונית תתבצע באתר הקורס ותכיל את הקבצים הבאים:
  - הקובץ dry.pdf הכולל את התשובות לחלק היבש של התרגיל.
  - הקובץ ex3.sml הכולל את כל הפונקציות שנדרשתם לממש בחלק הרטוב
    - רשימת הפונקציות שטעינת הקובץ ex3.sml צריכה להניב:
      - T
      - ++
      - **
      - invertible
      - max_city
      - min_city
  - הקובץ ex3.kt הכולל את כל הפונקציות שנדרשתם לממש בחלק (ממש) רטוב
    - רשימת הפונקציות שטעינת הקובץ ex3.kt צריכה להניב:
      - completeMe
  - אם עשיתם, תגישו את הבונוס תחת ספריה בשם lib_bonus.

לכל קובץ שמכיל קוד:

- הוסיפו בשורה הראשונה הערה המכילה את השם, מספר ת.ז. וכתובת המייל של המגישים מופרדים באמצעות רווח.
- וודאו שהקבצים נטענים/מתקמפלים ללא שגיאות גם לאחר הוספת ההערה שלעיל.

יש להגיש את הקבצים דחוסים בתוך קובץ zip. הקבצים יהיו בשורש קובץ ה-zip ולא בתוך ספרייה.
שם הקובץ יהיה EX2_ID1_ID2.zip כאשר 2ID1,ID הם מספרי ת.ז. של המגישים.
שימו לב שהבדיקה של החלק הרטוב היא אוטומטית, ולכן הקפידו על מילוי כל ההוראות בשביל למנוע בעיות מיותרות.

- על החלק היבש להיות מוקלד, אין להגיש סריקה או צילום של התשובות לחלק זה.
- אין צורך להגיש ניירת הסמסטר. תא הקורס לא יבדק במהלך הסמסטר, אז אנא חסכו בנייר.
- בודקי התרגילים מאוד אוהבים Memes. לאור ההצלחה בתרגילים הקודמים, גם הפעם, שתפו את תחושותיכם במהלך פתירת התרגיל באמצעות Meme מתאים על דף השער בהגשה - אולי יצא מזה משהו מעניין!

בהצלחה!

תיקונים והבהרות

חלקים רטובים

ניתן להניח שהקלטים תקינים.

חלק 2.1 - ML Lists - חימום

בחלק זה כלל המטריצות הן של מספרים ממשיים. כמו כן, המטריצות הן מטריצות, ואורך כל רשימה פנימית יהיה זהה.

4) הכפל הינו כפל מטריצות ולא איבר איבר.

חלק 2.2 - ספר מה ראית, ואבנה לך עיר

שטח בנוי יכול להיות בעל 0 קומות - נניח פארק, גינה, סתם מגרש וכו'.
הקלטים של הפונקציות הן:

max_city west north
min_city west north

שתי הרשימות לא יהיו ריקות.
נוספו ציורים כדי להבהיר את מבנה הערים הנדרשות.