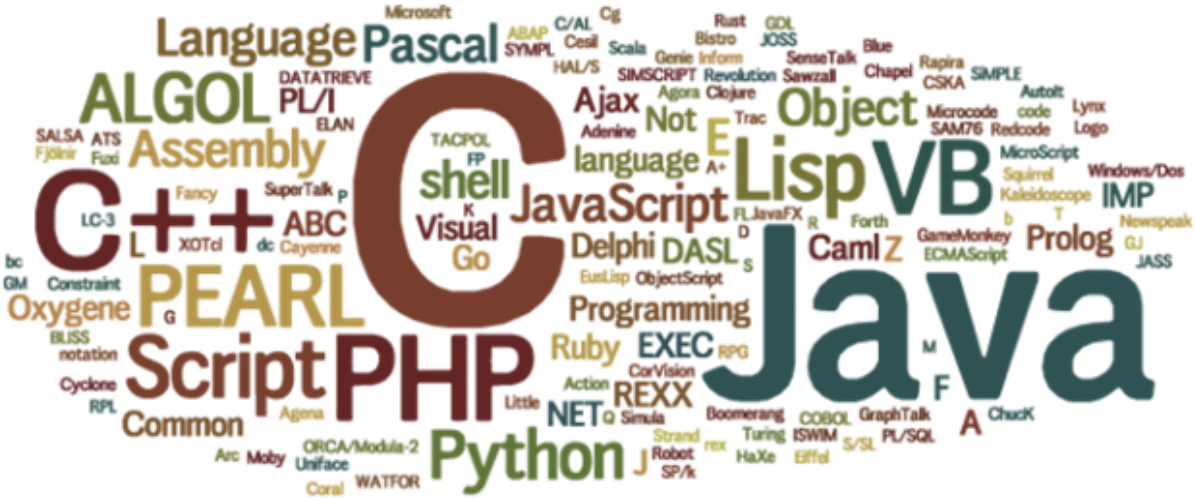
# שפות תכנות, 234319

חורף תשע"ח




# תרגיל בית 5

*ML sequences, Storage, Commands*

תאריך פרסום: 1.1.2018
מועד אחרון להגשה: 15.01.2018
מועד אחרון להגשה מאוחרת: 18.01.2018
מתרגל אחראי: ליאור בן-ימין
אי-מייל: lior.b@cs.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (**subject**) יהיה "**EX-PLW17**5" (ללא המרכאות).

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

תיקונים והבהרות יפורסמו בסוף מסמך זה, אנא הקפידו להתעדכן לעתים תכופות.

# חלק יבש

## שאלה 1 - RTTI

1. **קראו את [פרק 5.6 בחוברת השקפים](#) בנושא RTTI.**
   **מהו מנגנון RTTI? סכמו בשני משפטים.**

המנגנון RTTI, זהו רפרנס לתיאור הטיפוס המשותף לכל הערכים של הטיפוס (אורך, האם מחולק והדרך לפענח כל חלק):

- Dynamic typing – בדיקות טיפוס בזמן ריצה.
- Dynamic / static typing -איסוף אשפה, deep cloning.

2. **האם בשפת C יש מנגנון RTTI?**
   **אם כן - תארו אותו.**
   **אם לא - הסבירו למה.**

בשפת C אין מנגנון RTTI בגלל עיקרון ה"אין מחיר נסתר" בשפה. מכיוון שה-RTTI הוא מנגנון הפועל "מאחורי הקלעים", ה-RTTI נוגד את עקרונות השפה ב-C.

3. **הסבירו: כיצד מנגנון איסוף אשפה משתמש במנגנון RTTI?**

מנגנון RTTI שומר מידע מהיכן המשתנה הגיע ומי מחזיק רפרנס אליו. ללא מנגנון ה-RTTI לא נוכל לדעת אילו משתנים הם אשפה ואילו לא, ולא נוכל לאסוף אותם לאשפה.

4. **תארו שימוש אחר (שאינו איסוף אשפה) למנגנון RTTI.**

# שאלה 2 - KOTLIN

ענו על הסעיפים הבאים בהקשר של שפת Kotlin:

1. מהם הפקודות האטומיות בשפה?

2. מהם בנאי הפקודות בשפה? לאילו בנאים תיאורטים הם מתאימים?

3. מהם ה-Sequencers של השפה?

בדומה ל++C/C קיימות הפקודות: Return,Break,Continue המשמשות לשינוי רצף התוכנית באופן דומה בKotlin.

בנוסף בKotlin קיים הסימון "@" היכול להיות מצורף לכל ביטוי בKotlin כך שהוא יהפוך להיות "Label" שיהיה באפשרות המתכנת ל"קפוץ" למקום הזה ע"י continue@/break@:

```
loop@ for (i in 1..100) {
    for (j in 1..100) {
        if (...) break@loop
    }
}
```

בנוסף יש return@ בפונקציות למדה(lambda function), באופן הבא:

```
fun foo() {
    ints.forEach lit@{
        if (it == 0) return@lit
        print(it)
    }
}
```

קיימת גם מערכת ניהול שגיאות(Exception) בדומה ל++C ע"י המילים השמורות try/catch/finally.

4. מהם סוגי המערכים הקיימים בשפה? מה קורה בחריגה מגבולות מערך?

5. האם קיימים מנגנוני איסוף אשפה בשפה? אם כן, פרטו על אחד מהם.

מנגנון הGarbage Collector בKotlin זהה לזה של שפת הJava.

# שאלה 3 - BASH

ענו על הסעיפים הבאים בהקשר של שפת bash:

1. סווגו את מערכת הטיפוסים שלה לפי הקריטריונים שנלמדו בתרגול ובהרצאה.

סווג את מערכת הטיפוסים של BASH לפי הקטגוריות הבאות:

- Existance: כל המשתנים בשפה הם מאותו סוג: מערך של מחרוזות. לכן אין ב BASH מערכת טיפוסים, כלומר השפה היא Untyped. (הערה: נשים לב שפעולות אריתמטיות על משתנים אפשרית רק כאשר המחרוזת היא מספר, אך מחרוזות כאלו אינן מהוות טיפוס נפרד. כלומר הסמנטיקה נקבעת לפי הערך ולא לפי הטיפוס).
- Strength: מכיוון שאין מערכת טיפוסים (כל המשתנים מאותו טיפוס) אז אין צורך באכיפה, או בהמרות מסוימות לפני ביצוע פעולות כלשהן בין משתנים. התנהגות כזאת מכונה Weak Typing.
- Time of checking: השפה מאופיינת ב Dynamic Typing. כפי שהוזכר לעיל אין מערכת טיפוסים, לכן אין צורך לבדוק טיפוסיות של משתנים לפני ריצה. בזמן ריצה תיווצר שגיאה כאשר ננסה להפעיל פעולות אריתמטיות על מחרוזות שאינן מייצגות מספרים.
- Type Equivalence: קיים רק טיפוס אחד לכן כל הערכים שקולים מבחינת הטיפוס.
- Flexibility: כתוצאה מהעדר טיפוסיות למתכנת יש חופש פעולה רחב מאוד.

2. חזרו על שאלה 2 עבור bash.

2.1 -

2.2 -

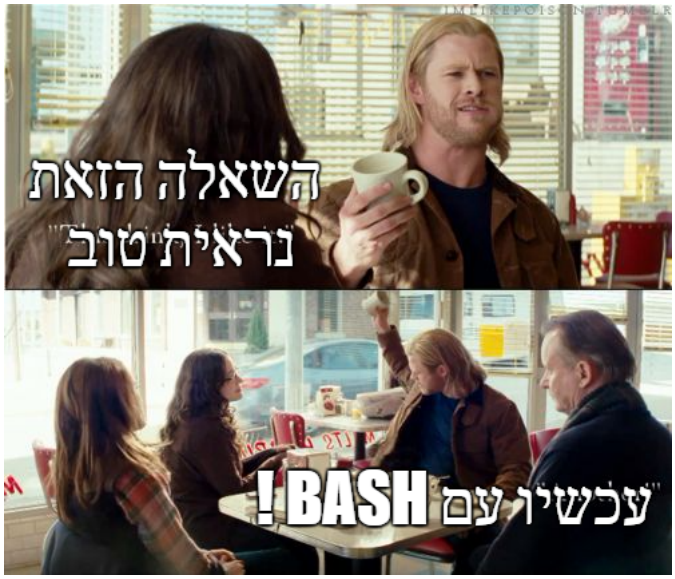
2.3 – בBash יש continue וגם break. בנוסף יש אפשרות לשנות את כמות האיטרציות של לולאה ע״י "continue N".
עבור חזרה מפונקציה קיים השימוש בreturn בדומה ל++C/C.

2.4 -

2.5 – לשפת Bash אין Garbage Collector.

3. כיצד נקבעת נקודת התחילה של התכנית בשפת bash? מהם "גבולות" התכנית? האם השפה אוטרקית?

נקודת ההתחלה של קטע קוד בשפת BASH היא הפקודה הראשונה בקובץ. באותו אופן, סוף התכנית מוגדר להיות הפקודה האחרונה בקובץ - אלו הם גבולות התכנית. משום שגבולות אלו מצויים בקובץ אחד ויחיד השפה עונה על ההגדרה של שפה אוטרקית - שפה שבה ניתן לדעת בדיוק איזה קוד יתבצע מהסתכלות בתכנית בלבד. כמו כן, אין צורך לייבא ספריות חיצוניות על מנת להשתמש בפונקציות קלט\פלט שונות אשר מובנות בשפה, כמו echo.


השאלה הזאת
נראית טוב
"עכשיו עם BASH!"

# 2. 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:

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

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:

   a, 'b) heterolist') <-build4 : 'a*'b*'a*'b

   שמחזירה רשימה מתחלפת, המחזיקה בדיוק את ארבעת הערכים שהועברו בפרמטר.
   הפתרון צריך להינתן בביטוי בודד (אין להגדיר פונקציות 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):

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

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:

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

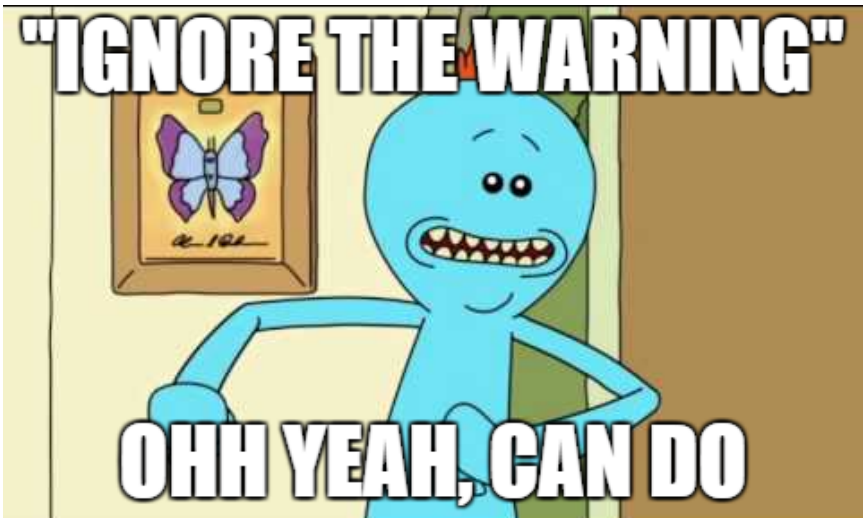
5. הגדירו פונקציה `a, 'b) heterolist') <-zip : 'a list * 'b list`

המבצעת את הפעולה ההפוכה לפונקציה בסעיף הקודם. במקרה שמספר האיברים ברשימות איננו מתאים (אם לא מתקיים ש□□□ len(b) ≤ len(a ≥ 1 -len(a) ) יש לזרוק את החריגה Empty. הפונקציה `id_heterolist` להלן (שאיננה להגשה) שקולה לפונקציית הזהות על `('a, 'b) heterolist`:

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

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

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



## שאלה 2: רצפים

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

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

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

```
;exception EmptySeq
;x,_)) = x | head Nil = raise EmptySeq)fun head(Cons
;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
          ;(a bseq' <-bCons of 'a * (direction  |
```

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
xf(Forward) | bForward bNil = raise  = ((fun bForward(bCons(_,xf
;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 `int bseq <-intbseq : int`, 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 -
int bseq : (val it = bCons (3,fn
;(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 -
bCons (4,fn) : int bseq = val it
;(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 -
bseq val it = bCons (1,fn) : int
;(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:

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

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 -
it = bCons (2,fn) : int bseq val
;(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 `a bseq' <-a seq ' <-seq2bseq : 'a seq`, 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:

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

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-
fn) : int bseq,0) val it = bCons
```

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 -
al it = bCons (3,fn) : int bseqv
;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
```

# הנחיות הגשה

- בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד (כולל) תרגול 9. אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.

- שימו לב כי ML היא sensitive-case, ודאו על כן כי שמות הפונקציות הן בדיוק כפי שנדרש בתרגיל, כמו כן יש לדאוג להסתיר את פונקציות העזר שאתם מממשים באמצעות let/local.

- אנא ודאו כי החתימות של הפונקציות שאתם מממשים זהות לחתימות הנתונות בסעיפים.

- ההגשה האלקטרונית תתבצע באתר הקורס ותכיל את הקבצים הבאים:
  - הקובץ **dry.pdf** הכולל את התשובות לחלק היבש של התרגיל.
  - הקובץ **ex5.sml** הכולל את התשובות לחלק הרטוב של התרגיל.

- 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 EX5_ID1_ID2.zip כאשר 2ID1,ID הם מספרי ת.ז. של המגישים.

- שימו לב שהבדיקה של החלק הרטוב וצורת ההגשה היא אוטומטית, ולכן הקפידו על מילוי כל ההוראות בשביל למנוע בעיות מיותרות.

- 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

- בשאלה 2 סעיף 5, הניסוח תוקן.