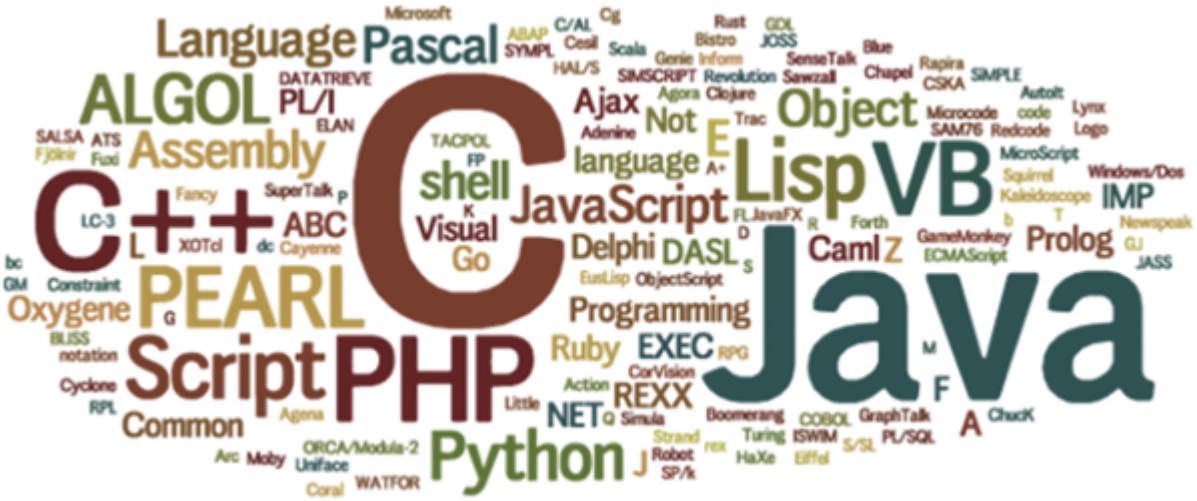
שפות תכנות, 236319

פרופ' י. גיל

חורף 2020-2021



תרגיל בית 3

תאריך פרסום: 24.11.20
מועד אחרון להגשה: 8.12.20
מועד אחרון להגשה מאוחרת: 12.12.20
מתרגל אחראי: מקסים ברסקי
אי-מייל: maxim.barsky@cs.technion.ac.il
בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PL-EX3" (ללא המירכאות).

תרגיל בית זה מורכב משני חלקים, חלק יבש וחלק רטוב.
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

קראו את התיעוד של שפת התכנות D וענו על הסעיפים הבאים:

1. מהם הטיפוסים הפרימיטיביים בשפה?
2. מהם בנאי הטיפוסים בשפה? לאילו בנאי טיפוסים תיאורטיים הם קשורים?
3. האם קיימים בנאים לא אורתוגונליים?
4. מהי שקילות הטיפוסים הנהוגה בשפה?

שאלה 2

הסבירו מהו quine והביאו דוגמה לאחד כזה.
בונוס: הסבירו למה quine יכול להיות רלוונטי לוירוסים.

שאלה 3

ענו על הסעיפים הבאים בשפת מיני-ליספ:

1. ממשו את הפונקציה nset, שדומה מאוד לפונקציה set מלבד שני הדברים הבאים:
   a. היא תמיד מחזירה NIL.
   b. לא צריך לעשות quote על הארגומנט הראשון שלה.
- להנחיות נוספות- תסתכלו על תרגילים 15-16 בספר הקורס.

הסעיף הבא הינו רשות, תשובה נכונה תקנה בונוס לציון תרגיל הבית (עד 10 נקודות)

2. מצאו ביטויים A,B כך ש:
   a. A != B.
   b. eval(A)= B.
   c. eval(B) = A.

חלק רטוב

בחלק זה עליכם להסתיר את פונקציות העזר שבהם השתמשתם באמצעות let ו-local.

שאלה 1

הגישו את הפתרון לשאלה זו בקובץ hw3_q1.sml

תזכורת: 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:

```
datatype ('a, 'b) heterolist = _________
      | _________ ;
infixr 5 ::: ;
```

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

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]
```

2. הגדירו את הפונקציה:

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

שתחזיר 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. <u>סעיף זה לא להגשה (אך משמש בשבילכם לבדיקת הטיפוסים שלכם):</u> 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)
```

4. הגדירו פונקציה אשר מקבלת רשימה מתחלפת ומפצלת אותה לשתי רשימות "רגילות", רשימה אחת של האיברים מהטיפוס הראשון ב-heterolist ורשימה שנייה של האיברים מהטיפוס השני ב-heterolist. אין לזרוק חריגות בפונקציה זו.

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

<u>דוגמאות הרצה:</u>

```
- unzip (build4(236, "hello", 319, "world"));
val it = ([236, 319], ["hello", "world"]) : int list * string list
```

5. הגדירו פונקציה zip אשר מבצעת את הפעולה ההפוכה לפונקציה בסעיף הקודם. כלומר הפונקציה תקבל שתי רשימות באורך זהה, וליצור מהן רשימת heterolist המורכבת מאיברי שתי הרשימות, כאשר האיבר הראשון ב-heterolist יהיה האיבר הראשון מהרשימה הראשונה והאיבר השני יהיה האיבר הראשון מהרשימה השנייה.

אם מספר האיברים בשתי הרשימות שונה, יש לזרוק את החריגה Empty.

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

<u>דוגמאות הרצה:</u>

```
- zip ([1, 2, 3, 4, 5], [#"a", #"b", #"c", #"d", #"e"]);
val it = 1::: #"a":::2:::#"b":::3:::#"c":::4:::#"d":::5:::#"e":::
<your “nil” value> : (int, char) heterolist

- zip ([1, 2], [#"a"]);
uncaught exception Empty;
```

שאלה 2

הגישו את הפתרון לשאלה זו בקובץ hw3_q2.sml

בשאלה זו נרצה לממש את אלגוריתם Knuth לאיזון וקטורים בינאריים. כלומר בהינתן וקטור של ביטים (המיוצג ע"י רשימה המכילה את המספרים 1 ו-0), נחזיר וקטור חדש אשר מכיל מספר זהה* של ביטים דולקים - בעלי הערך 1 וביטים כבויים - בעלי הערך 0.

אלגוריתם knuth לאיזון וקטורים פועל באופן הבא:

בהינתן וקטור (רשימה) של ביטים, נרצה לקודד וקטור שמספר הביטים בעלי הערך 1 בפלט יהיה זהה למספר הביטים בעלי הערך 0 בפלט. וכן, נוכל לפענח את הוקטור המקורי שקודדנו.

האלגוריתם פועל באופן הבא:

התחל באינדקס הראשון בוקטור ועבור עליו ביט-ביט עד שמספר הביטים בעלי הערך 1 שווה למספר הביטים בעלי הערך 0.

הפוך את הביט הנוכחי (אם הביט 1 אז רשום 0, ולהפך).

כאשר מספר הביטים הדלוקים שווה למספר הביטים הכבויים, שרשר את הקידוד הבינארי של האינדקס בו עצרנו את התהליך (האינדקס אחרי הביט האחרון שהפכנו) בסוף הרשימה.

* נשים לב כי קידוד האינדקס שבו עצרנו לא בהכרח מאוזן, לשם השאלה, נניח כי הוקטור מאוזן וכי אורך האינדקס זניח ולכן אין צורך שמספר הביטים שדלוקים בו יהיה שווה למספר הביטים הכבויים).

<u>יש להניח כי כל הוקטורים באורכים זוגיים</u>

1. תחילה, ממשו את הפונקציה to_binary אשר מקבלת מספר ממשי שלם אי שלילי ומחזירה רשימה המכילה את הקידוד הבינארי הקצר ביותר של המספר כאשר האיבר הראשון ברשימה הוא ה-lsb (כלומר הביט שערכו הקטן ביותר). עבור המספר 0 יש להחזיר רשימה ריקה.

<u>חתימת הפונקציה:</u>

```
val to_binary = fn : int -> int list;
```

<u>דוגמאות הרצה:</u>

```
- to_binary 5;
val it = [1, 0, 1] : int list;
- to_binary 6;
val it = [0, 1, 1] : int list;
- to_binary 0;
val it = [] : int list;
```

2. כעת, ממשו את הפונקציה encode אשר מקודדת וקטורים לפי אלגוריתם knuth (כלומר בהינתן וקטור, מחזירה וקטור שמספר הביטים הדלוקים בו שווה למספר הביטים הכבויים למעט האינדקס בסוף):

חתימת הפונקציה:

```
val encode = fn : int list -> int list;
```

דוגמאות הרצה:

```
- encode [1, 1, 0, 1];
val it = [0, 1, 0, 1, 1] : int list;
- encode [1, 1, 1, 1, 1, 1];
val it = [0, 0, 0, 1, 1, 1, 1, 1] : int list;
```

3. כעת לאחר שקודדנו את הוקטורים, נרצה לפענח אותם, ממשו פונקציה המקבלת וקטור מאוזן ומחזירה את הוקטור המקורי. כך שיתקיים `decode (encode x) = x`.

הפונקציה מקבלת את הוקטור המאוזן ואת אורך הוקטור המקורי שאוזן.

חתימת הפונקציה:

```
val decode fn : int list * int -> int list;
```

דוגמאות הרצה:

```
- decode ([0, 0, 0, 1, 1, 1, 1, 1], 6);
val it = [1, 1, 1, 1, 1, 1]
- decode ([0, 1, 0, 1, 1], 4);
val it = [1, 1, 0, 1]
```

שאלה 3

הגישו את הפתרון לשאלה זו בקובץ hw3_q3.sml

בתרגיל זה נממש טבלת ערבול (hash table) המשתמשת ב-open addressing.

<u>תזכורת:</u>

טבלת ערבול היא מבנה נתונים יעיל הממש מילון בצורה יעילה כאשר אין צורך לעבור על המילון לפי סדר המפתחות. חיפוש האיברים במילון מתבצע באמצעות פונקציית ערבול (hash).

טבלת הערבול ממומשת בתור מערך של זוגות של מפתח וערך, כאשר האינדקס שבו יישמר כל איבר נקבע לפי תוצאת הערבול על המפתח המתאים לאיבר. כלומר עבור המפתח והערך $(k\,,v)$, האיבר יישמר באינדקס $hash(k)$ במערך: $hashtable[hash(k)] \leftarrow (k,v)$

בהכנסת איבר חדש לטבלה, במידה והתרחשה התנגשות (כלומר התא במערך שמתאים למפתח כבר מכיל איבר אחר) נפתור את הבעיה בעזרת open addressing.
במימוש שלנו, אם התא ה-$hash(k) = i$ תפוס, אז נפעיל שוב את פונקציית הערבול, כלומר נחשב $hash(i)$ וננסה להכניס את האיבר לאינדקס זה. במידה וגם תא זה תפוס, נפעיל שוב את פונקציית הערבול על התוצאה עד שנקבל תא פנוי, או שטבלת הערבול מלאה ואז לא נוכל להכניס את האיבר.

על מנת למצוא איבר בטבלה, נחשב את האינדקס בו הוא אמור להימצא ע"י פונקציית הערבול על המפתח, נסמן את האינדקס $hash(k) = i$. נשווה את המפתח שאנחנו מחפשים והמפתח של האיבר שבאינדקס, במידה והמפתחות שווים מצאנו את האיבר המבוקש. אחרת, התרחשה התנגשות ונצטרך לחפש בתא הנמצא באינדקס $hash(i)$, אם המפתח בתא זה שווה למפתח שאנחנו מחפשים אז מצאנו כאן את האיבר שחיפשנו, אחרת נמשיך לחפש באופן רקורסיבי ע"י הפעלת פונקציית הערבול.
במידה והגענו לתא ריק, האיבר לא נמצא בטבלה.

מחיקה בטבלת הערבול תתבצע בצורה מעט מסובכת, משום שאם פשוט נוציא איבר מהטבלה לא נוכל למצוא את האיבר שמתנגש איתו שבא אחריו בשרשרת הערבול.
לדוגמא: עבור האיברים (k_1, v_1), (k_2, v_2) שמתנגשים בערבול, כלומר מתקיים $hash(k_1) = hash(k_2)$. נניח שתחילה הוכנס (k_1, v_1) באינדקס $hash(k_1) = i$ ואחריו הוכנס (k_2, v_2) באינדקס $hash(i) = hash(hash(k_2))$ בשל ההתנגשות. אם נוציא את האיבר (k_1, v_1) שנמצא באינדקס i, כאשר נחפש את (k_2, v_2) נבדוק תחילה באינדקס i ונמצא כי התא פנוי, משום שהוצאנו אותו מהטבלה, ולכן נפסיק את החיפוש ונחזיר שהאיבר לא נמצא בטבלה.
על מנת לפתור בעיה זו, במחיקת איבר, נסמן אותו כמחוק. כשנחפש איברים בטבלה, כאשר נגיע לאיבר מחוק נמשיך לחפש כאילו שתא זה לא ריק. כאשר נכניס איברים לטבלה וניתקל בתא מחוק, נוכל להכניס אליו ערך כאילו שהוא היה תא ריק.

<u>מימוש:</u>

תחילה עליכם להגדיר את טיפוסי העזר הבאים:

```
datatype 'a HashTableEntry = Empty
    | Deleted
    | Value of int * 'a;
```

טיפוס זה יהיה הטיפוס של הערכים בטבלת הערבול, כל כניסה בטבלה תהיה אחת משלושת הערכים האפשריים. תא ריק שעוד לא הוכנס אליו ערך, תא מחוק שהוכנס אליו ערך ונמחק או תא המכיל ערך, כאשר תא שמכיל ערך מכיל זוג של מפתח מטיפוס `int` וערך מטיפוס `'a`.

בנוסף הגדירו את ה-alias הבא:

```
type 'a HashTable = 'a HashTableEntry list;
```

זהו שם ברור יותר לטיפוס האמיתי של טבלת הערבול שלנו. הטיפוס של טבלת הערבול הוא רשימה של HashTableEntry.

1. ממשו את הפונקציה create המקבלת מספר n ומחזירה טבלת ערבול ריקה בגודל n, שכל איבריה הם Empty.

שימו לב: בזמן יצירת הפונקציה יש לרשום במפורש את הטיפוס של טבלת הערבול. לדוגמה:

```
val htable : int HashTable = create 8;
```

<u>חתימת הפונקציה:</u>

```
val create = fn : int -> 'a HashTableEntry list
```

<u>דוגמאות הרצה:</u>

```
- create 8;
val it = [Empty,Empty,Empty,Empty,Empty,Empty,Empty,Empty]
```

<u>פונקציית הערבול:</u>

נגדיר את פונקציית הערבול להיות מהטיפוס `int -> int`, הפונקציה תקבל מספר, שהוא המפתח ותחזיר מספר שהוא האינדקס ברשימה שבו אמור להימצא האיבר המתאים למפתח. יש להניח כי האינדקס המוחזר מפונקציית הערבול הוא אינדקס חוקי בטבלת הערבול, כלומר לא חורג מגודל הטבלה.

על מנת לבדוק את עצמכם בתרגיל, אתם יכולים להגדיר את פונקציית הערבול הבאה:

```
- fun h n x = (x + 1) mod n;
- val h = fn : int -> int -> int
```

על מנת להשתמש בפונקציה בטבלת ערבול מגודל n, נגדיר:

```
- val h = h n;
val it = fn : int -> int;
```

אין להגיש את הפונקציה h.

2. ממשו את הפונקציה insert אשר מקבלת בצורה curried את פונקציית הערבול, את טבלת הערבול וזוג (tuple) של המפתח והערך. הפונקציה תכניס את הזוג של המפתח והערך לטבלת הערבול, כלומר תחזיר טבלת ערבול חדשה שזהה לקיימת בתוספת הזוג שיש להוסיף, לפי האלגוריתם המפורט בסוף הסעיף.
עליכם להכניס לתא המתאים הראשון בטבלה, בין אם פנוי או מכיל איבר בער מפתח זהה (במקרה זה עליכם להחליף אותו).

חתימת הפונקציה:

```
val insert = fn : (int -> int) -> 'a HashTable -> int * 'a -> 'a
HashTable
```

<u>דוגמאות הרצה:</u>

```
- val htable : int HashTable = create 8;
val it = [Empty,Empty,Empty,Empty,Empty,Empty,Empty,Empty]
- val htable = insert h htable (3, 6);
val it = [Empty, Empty, Empty, Empty, Value(3,6), Empty, Empty,
Empty]
- val htable = insert h htable (5, ~1);
val it = [Empty, Empty, Empty, Empty, Value(3,6), Empty,
Value(5,~1), Empty]
- val htable = insert h htable (11, 2);
val it = [Empty, Empty, Empty, Empty, Value(3,6), Value(11,2),
Value(5,~1), Empty]
```

ניתן להניח שתמיד יש מקום להכניס איברים לטבלת הערבול.

3. ממשו את הפונקציה get אשר מקבלת בצורה curried את פונקציית הערבול, את טבלת הערבול ואת המפתח של הערך שיש למצוא. הפונקציה תחזיר את הערך המתאים למפתח.

<u>חתימת הפונקציה:</u>

```
val get = fn : (int -> int) -> 'a HashTable -> int -> int * 'a
```

<u>דוגמאות הרצה:</u>

```
<continues previous example>
- get h htable 11;
val it = (11,2)
- get h htable 3;
val it = (3, 6)
- get h htable 5;
val it = (5,~1)
```

ניתן להניח כי האיבר שמחפשים נמצא בטבלה.

4. ממשו את הפונקציה remove אשר מקבלת בצורה curried את פונקציית הערבול, את טבלת הערבול ואת המפתח של האיבר שיש למחוק. הפונקציה תמחק את האיבר שמתאים למפתח, כלומר תחזיר טבלת ערבול חדשה ללא האיבר שרצינו למחוק.

חתימת הפונקציה:

```
val remove = fn : (int -> int) -> 'a HashTable -> int -> 'a
HashTable
```

דוגמאות הרצה:

```
- val htable: int HashTable = create 8;
val it = [Empty,Empty,Empty,Empty,Empty,Empty,Empty,Empty]
- val htable = insert h htable (0, 3);
val htable = [Empty,Value (0,3),Empty,Empty,Empty,Empty,Empty,Empty]
- val htable = insert h htable (3, ~2);
val htable = [Empty,Value (0,3),Empty,Empty,Value
(3,~2),Empty,Empty,Empty]
- val htable = remove h htable 0;
val htable = [Empty,Deleted,Empty,Empty,Value
(3,~2),Empty,Empty,Empty]
- val htable = insert h htable (11, 5);
val htable = [Empty,Deleted,Empty,Empty,Value (3,~2),Value
(11,5),Empty,Empty]
- val htable = insert h htable (19, 0);
val htable = [Empty,Deleted,Empty,Empty,Value (3,~2),Value
(11,5),Value (19,0),Empty]
- val htable = remove h htable 11;
val htable = [Empty,Deleted,Empty,Empty,Value (3,~2),Deleted,Value
(19,0),Empty]
```

```
get h htable 19;
val it = (19, 0)
```

ניתן להניח שהאיבר שרוצים למחוק נמצא בטבלה.

הנחיות

- בתרגיל זה ניתן להשתמש רק בחומר שנלמד בשפת ML עד (וכולל) תרגול 6. אין להשתמש באף פונקציה או תכונה של השפה שלא נלמדה בתרגולים.
- רשימת הקבצים שצריכים להופיע בתוך קובץ ה-zip היא:
  `dry.pdf, hw3_q1.sml, hw3_q2.sml, hw3_q3.sml`
- בכל קובץ קוד, הוסיפו בשורה הראשונה הערה המכילה את השם, מספר ת"ז וכתובת הדואר האלקטרוני של המגישים מופרדים באמצעות רווח.
- על החלק היבש להיות מוקלד, אין להגיש סריקה או צילום של התשובות לחלק זה.
- שם קובץ ההגשה יהיה EX2_ID1_ID2.zip כאשר ID1, ID2 הם מספרי ת.ז. של המגישים בסדר עולה ממש. אם לא ניתן לסדר את מספרי ת"ז בסדר שכזה יש לפנות למשרד הפנים.
- אין צורך להגיש ניירת הסמסטר. תא הקורס לא יבדק במהלך הסמסטר, אז אנא חסכו בנייר.
- בודקי התרגילים מאוד אוהבים Memes. שתפו את תחושותיכם במהלך פתירת התרגיל באמצעות Meme מתאים על דף השער בהגשה - אולי יצא מזה משהו מעניין!

בהצלחה!

תיקונים והבהרות:

1. תיקון דוגמאות ההרצה בשאלה 2, סעיף 3.
2. דוגמאות ההרצה בכל השאלות עודכנו.
3. הערה בשאלה 3 בפונקציה insert.
4. הבהרה בדוגמה בשאלה 1 בסעיף 5 בחלק הרטוב.
5. חלק יבש, שאלה 3 סעיף 2 הינו בונוס.