חלק רטוב

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

- קבצים המסופקים לחלק זה תוכלו למצוא בגיטהאב של הקורס - Homework3.
- פתרון לכל אחת מהשאלות יש לכתוב בקובץ נפרד ששמו hw3_q<i>.sml עבור שאלה i. בנוסף לשאלות 1 ו-2 מסופקת לכם דוגמת קלט ופלט רצוי. לצורך העניין את הפתרון שלכם לשאלה 1 תוכלו לבדוק בעזרת הפקודות הבאות:

```
smlnj hw3_q1.sml < q1.in | sed '1,/===TEST START===/d' >
q1.out
diff q1.out q1.expected
```

- סיפקנו עבורכם סביבת פיתוח מוכנה לSML ב-VSCode. מדריך התקנה ושימוש.
- יש לבדוק, לקמפל ולהריץ את פתרונותיכם לתרגיל זה בסביבה הנ"ל.

שאלה 1

בשאלה זו נרצה לממש את האלגוריתם של 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.

האלגוריתם של 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. וכך, נוכל לפענח את הוקטור המקורי שקודדנו.

האלגוריתם פועל באופן הבא:

- התחל באינדקס הראשון בוקטור ועבור עליו ביט-ביט
- בכל צעד, הפוך את הביט הנוכחי (0 ל-1 ו-1 ל-0)
  - כאשר נגיע למצב של איזון **בכל הוקטור (#1s = #0s)**, שרשר את הקידוד הבינארי של האינדקס בו עצרנו את התהליך (האינדקס אחרי הביט האחרון שהפכנו) בסוף הרשימה

<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.
- האיבר הראשון ברשימה הוא ה-lsb (כלומר הביט שערכו הקטן ביותר).
- עבור המספר 0 יש להחזיר רשימה ריקה.

```
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 האלגוריתם של Knuth (כלומר בהינתן וקטור, מחזירה וקטור שמספר הביטים הדלוקים בו שווה למספר הביטים הכבויים למעט האינדקס בסוף):

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

<u>דוגמאות הרצה:</u>

```
- 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;
- encode [1, 0, 1, 1, 1, 1];
val it = [0, 1, 0, 0, 1, 1, 0, 0, 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.

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

<u>דוגמאות הרצה:</u>

```
- 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]
- decode ([0, 1, 0, 0, 1, 1, 0, 0, 1], 6);
val it =  [1, 0, 1, 1, 1, 1] : int list
```

שאלה 2

תזכורת: הטיפוס 'a list בשפת ML מוגדר כך:

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

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.

2. הגדירו את הפונקציה:

```
val build4 = fn : '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. 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.

```
val unzip = fn : ('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
```

4. הגדירו פונקציה zip אשר מבצעת את הפעולה ההפוכה לפונקציה בסעיף הקודם.
אם מספר האיברים בשתי הרשימות שונה, יש לזרוק את החריגה Empty.

```
val zip = fn : '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":::NIL :
(int, char) heterolist

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

שאלה 3

התרגיל הזה הוא צעד ראשון לקראת כתיבת מפרש לשפת Lisp בשפת ML. אחד מהשלבים הראשונים בפעולת המהדר (קומפילייר) או המפרש הוא פירוק הקוד ל-tokens.

token הוא יחידה בעלת משמעות תחבירית עבור המפרש. מכיוון שטרם למדתם קורס בקומפילציה, סיפקנו לכם את המימוש של tokenizer בקובץ test_parse.sml.
ה-tokenizer מפרק מחרוזות של S-Expressions ל-tokens, כאשר הפעלתו היא צעד מקדים בפרסור תוכניות של Lisp:

```
fun tokenize x =
String.tokens (fn c => c = #" ")
    (String.translate (fn #"(" => "( " | #")" => " )" | c => str c) x);
```

מימוש מפרש של Lisp ב-ML

בתרגיל זה נשתמש בפונקציה tokenize על מנת לבנות ביטוי (S-Expression) שניתן להזינו לפונקציה EVAL (שתראו בהרצאות - **מומלץ מאוד להגיע!** - ותממשו בהמשך בתרגילי הבית הבאים). כלומר, בתרגיל זה נממש פונקציה בשם parse שמקבלת כפרמטר רשימת tokens ומחזירה ערך שמייצג S-Expressions.

1. כפי שלמדתם בהרצאה, הערכים בשפת Lisp הם מטיפוס S-Expression, כאשר S-Expression הוא עץ בינארי שהעלים שלו הם אטומים. בשפת Lisp קיימים שני סוגי אטומים:
   - NIL - רשימה ריקה.
   - SYMBOL - שם סימבולי (לאו דווקא מקושר לערך בסביבה).

הוסיפו את הגדרת הטיפוס הבאה לקובץ הפתרון שלכם:

```
datatype Atom =
   NIL | SYMBOL of string
```

כעת נרצה להגדיר את הטיפוס SExp,

הוסיפו את הגדרת הטיפוס הבאה לקובץ הפתרון שלכם:

```
datatype SExp =
   ATOM of Atom | CONS of (SExp * SExp)
```

2. ממשו את הפונקציה:

```
val parse = fn : string list -> SExp
```

כאשר הפרמטר שלה הוא רשימת tokens המייצגת תוכנית, והערך המוחזר הוא התוכנית המפורסרת באופן הרקורסיבי הבא:

- מחרוזת s:

```
ATOM (SYMBOL s)
```

- רשימה `(x1 x2 … xn)`:

```
CONS (x1, CONS (x2, CONS (..., CONS(xn, ATOM NIL))))
```

ניתן להניח כי במקרה של רשימות - הסוגריים מאוזנים.

רמז: חשבו על המקרה הכללי לטיפול בסוגריים, וטפלו במקרה המיוחד של תחילת הפרסור ע"י pattern matching ברמת parse.

<u>דוגמאות הרצה:</u>

דוגמה 1:

```
CONS(
    ATOM (SYMBOL "+"),
    CONS(
        ATOM (SYMBOL "2"),
        CONS(
            ATOM (SYMBOL "3"), ATOM NIL
        )
    )
) = (parse (tokenize "(+ 2 3)"));
val it = true : bool
```

דוגמה 2:

```
ATOM (SYMBOL "a") = (parse (tokenize "a"));
val it = true : bool
```

דוגמה 3:

```
CONS(
    ATOM (SYMBOL "+"),
    CONS(
        ATOM (SYMBOL "1"),
        CONS(
            CONS(
                ATOM (SYMBOL "+"),
                CONS(
                    ATOM (SYMBOL "1"),
                    CONS(
                        ATOM (SYMBOL "3"), ATOM NIL
                    )
                )
            ), ATOM NIL
        )
    )
) = (parse (tokenize "(+ 1 (+ 1 3))"));
val it = true : bool
```

דוגמה 4:

```
CONS(
    ATOM (SYMBOL "define"),
    CONS(
        ATOM (SYMBOL "r"),
        CONS(
            ATOM (SYMBOL "5"), ATOM NIL
        )
    )
) = (parse (tokenize "(define r 5)"));
val it = true : bool
```

דוגמה 5:

```
CONS(
    ATOM (SYMBOL "+"),
    CONS(
        CONS(
            ATOM (SYMBOL "+"),
            CONS(
                ATOM (SYMBOL "1"),
                CONS(
                    ATOM (SYMBOL "3"), ATOM NIL
                )
            )
        ),
        CONS(
            ATOM (SYMBOL "1"), ATOM NIL
        )
    )
) = (parse (tokenize "(+ (+ 1 3) 1)"));
val it = true : bool
```

שימו לב שסיפקנו עבורכם קובץ "test_parse.sml" שמכיל סביבת טסטים לתרגיל.

בעת הרצת הקובץ, הטרמינל צריך להיות בתיקיה שמכילה את קובץ הפתרון שלכם "hw3_q3.sml". בקימפול הקוד אתם אמורים לקבל פלט של הודעות הצלחה:

```
Running Test single symbol... Passed
Running Test list with single element... Passed
Running Test empty list... Passed
Running Test nested lists... Passed
Running Test simple list... Passed
Running Test list with nested empty lists... Passed
Running Test deeply nested lists... Passed
Running Test complex expression... Passed
Running Test mixed atoms and lists... Passed
```

```
Running Test complex lambda... Passed
```

במקרה שתרצו להריץ את parse ישירות, הוסיפו את השורות הבאות לקוד שלכם:

```
Control.Print.printLength := 100;
Control.Print.printDepth := 100;
```

הנחיות הגשה

- את הפתרון לתרגיל הבית יש להגיש בקובץ zip המכיל את הקבצים הבאים בלבד:

hw3_q1.sml, hw3_q2.sml, hw3_q3.sml, dry.pdf

- על החלק היבש להיות מוקלד, אין להגיש סריקה או צילום של התשובות לחלק זה.
- שם קובץ ההגשה יהיה EX3_ID1_ID2.zip כאשר ID1, ID2 הם מספרי ת.ז. של המגישים.
- בודקי התרגילים **מאוד** אוהבים Memes. שתפו את תחושותיכם במהלך פתירת התרגיל באמצעות Memes מתאימים ב-pdf של החלק היבש. Memes מצחיקים בצורה יוצאת דופן יזכו את היוצרים בבונוס.

בהצלחה!