שפות תכנות תרגיל 1

שאלה 1

```
type element =
        | Z1
        | Z2 ;;
type sequence =
        | Empty
        | Cons of element * sequence ;;
let rec len = function
        | Empty -> 0
        | Cons (_, tail) -> 1 + len tail ;;

let rec append x y = match x with
        | Empty -> y
        | Cons (head, tail) -> Cons (head, append tail y) ;;
```

טענה:

$$\forall x, y \in sequence : len(x \; append \; y) = len(x) + len(y)$$

נוכיח את הטענה ע"י אינדוקציה מבנית.

מקרה בסיס:

תהיו $x, y \in sequence$ ונניח ש-$x = Empty$ אז לכל ערכי y מתקיים

$$len(x \; append \; y) =^1 len(y) = len(y) + 0 =^2 len(y) + len(x)$$

1. נשים לב שמתקיים

$$x \; append \; y = y$$

כיוון שלפי הגדרת של פונקציה $append$ הפעלה שלה על x, y כאשר $x = Empty$ מחזיר y.

2. לפי הגדרה של פונקציה len עבור $sequence$ מסוג $Empty$ היא מחזירה 0 ולכן $len(x) = Empty$.
3.

עכשיו לצורך ההוכחה נסמן את $x = head :: tail$ כאשר $head \neq Empty$ ו-$tail$ יכול להיות $sequence$ מכל סוג.
עכשיו נעשה הנחנה על $tails$ וצעד האינדוקציה נבצע על $x = head :: tail$.

> **Note**
>
> נציין שהשתמשנו בסימון "::" במשמעות ש-x היא רשימה שיש בה איבר ראשון head ושאר הרשימה tail.
> למרות שב-OCAML יש צורה כתיבה הזאת לפקודה append, אין כאן כוונה בה.

הנחה:

נניח שלכל $x, y \in sequence$ כך ש-x מצורה $x = tail$

$$len(tails\ append\ y) = len(tail) + len(y)$$

צעד:

נראה שלכל $x, y \in sequence$ כך ש-x מצורה $head :: tail$ הטענה מתקיימת.

$$len(x\ append\ y) =^1 len(Cons(head, tail\ append\ y)) =^2$$

$$= 1 + len(tail\ append\ y) =^3 1 + len(tail) + len(y) =^4 1 + len(x) - 1 + len(y) =$$

$$= len(x : tails) + len(y)$$

1. לפי הגדרה של פקודה $append$
2. לפי הגדרה של פקודה len
3. לפי הנחנת האינדוקציה
4. לפי הביטוי הבא:

$$len(head :: tails) = 1 + len(tails) \implies len(tails) = len(head :: tails) - 1$$

כאשר שיוויון ראשון לפי הגדרה של פקודה len.

מכאן הראנו שהטענה מתקיימת. מ.ש.ל.

שאלה 3 (שניה לפי הסדר הופעתה)

```
type 'a btree =
        | Empty
        | Node of 'a * 'a btree * 'a btree;;
```

```
let rec height = function
        | Empty -> 0
        | Node(_, left, right) -> 1 + max (height left) (height right);;
```

טענה

לכל עץ בינארי t מסוג $'a\ btree$ מתקיים כי האורך של t גדול או שווה מאורך המסלול הארוך ביותר בין שורש העץ לאחד העלים שלו.

ניזכר שאורך המסלול בין שני הקודקודים מוגדר להיות כמות הקשתות שמופיעות במסלול הזה. נסכים שעוב עץ ריק אורך המסמלול הוא 1- כיוון שלא קיימים קודקודים בעץ אז גם לא קיים שום מסלול בעץ, בנוסף אורך המסלול בעץ עם קודקוד אחד הוא 0 והמסלול היחיד שקיים הוא מקודקוד לעצמו.

נסמן את האורך המקסימלי בעץ של עץ ע"י $len_M(t)$ ואת הגובה של עץ המוגדר ע"י פונקציה הנתונה לעיל נסמן בהתאם $height(t)$.

נשים לב שגם לכל עץ לא ריק מתקיים

$$len_M(tree) = 1 + max\{len_M(tree.left), len_M(tree.right)\} \qquad (1)$$

אחרי שתיאמנו את ההגדרות וסימונים נעבוד להוכחה של טענה. נוכיח אותה באינדוקציה מבנית.

מקרה בסיס:

יהיה t עץ מסוג $'a\ btree$ כך שמתקיים $t = Empty$ אזי מקתיים

$$height(t) =^1 0 \geq -1 = len(t)$$

1. המעבר הזה נובע מהגדרה של פונקצית height כי עבור עץ ריק היא מחזירה 0.

Note

נשים לב שאפילו אם מישהו היה מגדיר את האורך המקסימלי של עץ ריק להיות 0 זה עדיין היה מתקיים.

הנחה:

יהיו $t_1, t_2 \in a'\ btree$ כלומר ונניח שמתקיים עבורם

$$height(t_1) \geq len_M(t_1), \quad height(t_2) \geq len_M(t_2)$$

צעד האינדוקציה:

נוכיח עבור $Node(\_, t_1, t_2)$.

$$height(Node(t_1, t_2)) =^1 1 + max(height(t_1), height(t_2)) \geq^2$$

$$\geq 1 + max(len_M(t_1), len_M(t_2)) =^3 len_M(Node(t_1, t_2))$$

1. לפי הגדרה של פונקציית height
2. לפי הנחת האינדוקציה
3. לפי אי שיוויון (1)

מכאן הטענה הין מתקיימת.

שלאה 4 (שלישית לפי הסדר הופעתה)

```
type bool_expr =
        Var of string
        Not oof bool_expr
        And of bool_expr * bool_expr
        Or of bool_expr * bool_expr;;
```

```
let rec num_of_vars = fun exp -> match exp with
        | Var(_) -> 1
        | And(x,y) -> (num_of_vars x) + (num_of_vars y)
        | Or (x,y) -> (num_of_vars x) + (num_of_vars y)
        | Not(x) -> (num_of_vars x);;
```

```
let rec num_of_connectives = fun exp -> match exp with
        | Var(_) -> 0
        | And(x,y) -> (num_of_connectives x) + (num_of_connectives y) + 1
        | Or(x,y) -> (num_of_connectives x) + (num_of_connectives y) + 1
        | Not(x) -> (num_of_connectives x) + 1;;
```

א. הוכיחו באינדוקציה מבנית או הפריכו באמצעות דוגמא נגדית: לכל ביטוי exp מטיפוס bool_expr מתקיים

```
nums_of_vars(expr)=num_of_connectives(expr) + 1
```

הפרכה:

נתבונן בביטוי

```
Not(X)
```

כאשר $X \in Var$ אזי מתקיים

```
num_of_vars Not(X) = (num_of_vars x) = 1
```

```
num_of_connectives (Not(X)) + 1 = 1 + num_of_connectives (X) + 1 = 1 + 0 + 1= 2
```

מכאן הטענה אינה נכונה.

ב.הוכיחו באינדוקציה מבנית או הפריכו באמצעות דוגמא נגדית: לכל ביטוי exp מטיפוס expr_bool בו לא מופיע Not מתקיים:

```
nums_of_vars(expr)=num_of_connectives(expr) + 1
```

הוכחה אינדוקציה

מקרה בסיס:

יהיה $X \in Var$ אז

```
num_of_vars (X) = 1
```

```
num_of_connectives(expr) + 1 = 0 + 1= 1
```

כל המעבר מוצדק ע"י הגדרות של פונקציות.

הנחה:

יהיו $X, Y \in expr$ ונניח שעבורם מתקיים

```
nums_of_vars(X)=num_of_connectives(X) + 1
nums_of_vars(Y)=num_of_connectives(Y) + 1
```

צעד:

נחלק למקרים ונוכיח עבור כל אחד מהם.
נוכיח עבור $And(X,Y)$

```
num_of_vars And(X,Y) = (num_of_vars X) + (num_of_vars Y) =
                        = num_of_connectives(X) + 1 + num_of_connectives(Y) + 1 =
                        = (num_of_connectives(X) + num_of_connectives(Y) + 1) + 1 =
                        = num_of_connectives(And(X,Y)) + 1
```

1. מעבר ראשון נובע מהגדרה של פונקציה num_of_vars ומעבר אחרון נובע מהדרה של פונקצייה num_of_connectives.
2. מעברר שני לפי הנחת האינדוקציה
3. מעבר שלישי קיבוץ איברים

וכיח עבור $Or(X,Y)$

```
num_of_vars Or(X,Y) = (num_of_vars X) + (num_of_vars Y) =
                        = num_of_connectives(X) + 1 + num_of_connectives(Y) + 1 =
                        = (num_of_connectives(X) + num_of_connectives(Y) + 1) + 1 =
                        = num_of_connectives(Or(X,Y)) + 1
```

1. מעבר ראשון נובע מהגדרה של פונקציה num_of_vars ומעבר אחרון נובע מהדרה של פונקצייה num_of_connectives.
2. מעברר שני לפי הנחת האינדוקציה
3. מעבר שלישי קיבוץ איברים

מכאן העטנה הינה נכונה.