שפות תכנות, 236319

פרופ' ד. לורנץ

אביב 2021



תרגיל בית 2

**תאריך פרסום**: 20/4/2021

**מועד אחרון להגשה**: 6/5/2021

**מועד אחרון להגשה מאוחרת**: 10/5/2021

**מתרגל אחראי**: יאיר טויטו

**אי-מייל**: twyair@campus.technion.ac.il

**בפניה בדוא"ל, נושא ההודעה (subject) יהיה "PL-EX2" (ללא המֵירכאות).**

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:

עבור כל אחת מהשפות הבאות קבע לאילו פרדיגמות היא שייכת והסבר בקצרה.

scala, rust, ruby, javascript, clojure, SPARQL, ATS

**Scala: functional, imperative**

**שפה מונחית עצמים כי כל ערך בה הוא עצם**

**שפה פונקציונלית כי כל פונקציה היא ערך ולכן אפשר להשתמש בה כמו ערך.**

**Rust:** [**concurrent**](https://en.wikipedia.org/wiki/Concurrent_programming)**,** [**functional**](https://en.wikipedia.org/wiki/Functional_programming)**,** [**generic**](https://en.wikipedia.org/wiki/Generic_programming)**,** [**imperative**](https://en.wikipedia.org/wiki/Imperative_programming)**,** [**structured**](https://en.wikipedia.org/wiki/Structured_programming)

**שפה שיש לה ביצועיות גבוהה ושמה דגש על בטיחות ומהירות.**

**Ruby:** [**functional**](https://en.wikipedia.org/wiki/Functional_programming)**,** [**imperative**](https://en.wikipedia.org/wiki/Imperative_programming)**,** [**object-oriented**](https://en.wikipedia.org/wiki/Object-oriented_programming)**,** [**reflective**](https://en.wikipedia.org/wiki/Reflective_programming)

**שפת מונחת עצמים טהורה. יש בה ביטויים רגלוריים. יש לה garbage collector.**

**Javascript*:*** [**functional**](https://en.wikipedia.org/wiki/Functional_programming)**,** [**imperative**](https://en.wikipedia.org/wiki/Imperative_programming)

**שפת תכנות מתאימה לאתרי אינטרנט ורצה על ידי דפדפן האינטרנט בצד הלקוח.**

**השפה משתמשת במפרש. השפה תומכת בסינטקס של שפת C אך היא גם סוג של מונחת עצמים. השפה היא מונחת אירועים.**

**Clojure:** [**concurrent**](https://en.wikipedia.org/wiki/Concurrent_programming)**,** [**functional**](https://en.wikipedia.org/wiki/Functional_programming)

**השפה מעודדת תכנות פונקציונלי ו"מקומפלת" ע"י על JVM.**

**SPARQL:** [**Query language**](https://en.wikipedia.org/wiki/Query_language)

**שפת שאילתות למסדי נתונים. פקודות בשפה זו נועדו לטפל בנתונים.**

**ATS:**[**functional**](https://en.wikipedia.org/wiki/Functional_programming)**,** [**imperative**](https://en.wikipedia.org/wiki/Imperative_programming)

**שפת תכנות שנועדה לאחד תכנות עם מפרט רשמי. מערכת הטיפוסים של ATS מאפשרת תכונות רבות שנחשבות "לא בטוחות" (למשל, הקצאת זיכרון) ,מה שהופך את ATS לשפת תכנות עם בטיחות ברמה נמוכה.**

שאלה 2:

tail-recursion (רקורסיית זנב)

1. הסבר מה זה tail-recursion.  
   **רקורסיית זנב היא פונקציה רקורסיבית שהיא מתכוננת, כך שהפקודה האחרונה של הפונקציה, זוהי הקריאה לפונקציה עצמה, ובעצם החישוב של הביטוי מסתיים בקריאה האחרונה לפונקציה. ז"א הפונקציה לא מתקפלת חזרה בעץ הרקורסיה אלא פשוט מחזירה ערך (בתחתית העץ).**

1. איך ניתן להשתמש ב-tail-recursion כדי ליעל תוכניות בשפות פונקציוליות?  
   **ניתן להשתמש בהם על ידי כך שנעשה את החישובי הנדרשים לפני הקריאה ונכוון את הביטוי כך שהפקודה האחרונה בפונקציה כזו , תהיה הקריאה לפונקציה עצמה, ובכך נחסוך בסיבוכיות מקום וזמן.**
2. כתוב ב-ML פונקציה שהיא tail-recursive המקבלת מספר טבעי n ומחזירה את n!.

**fun fact n =let**

**fun hlpfact 0 sum = sum**

**| hlpfact n sum = hlpfact(n-1,n\*sum)**

**in**

**hlpfact n 1**

**end;**

שאלה 3:

1. כתבו בקצרה את ההבדלים העיקריים בין BNF ל-EBNF.

**BNF -זה כללים בסיסיים להגדרת שפות**

**EBNF -מאפשר (Syntactic Sugar) BNF  זה הרחבה של**

1. כתבו EBNF קצר המגדיר את ליטרל המספרים: מספרים שלמים (למשל, 17), מספרים עם נקודה עשרונית (למשל, 63.77) ומספרים בכתיב מדעי  (למשל, 1.5E-8). הן חיוביים והן שליליים.

**number= [[sign], digit ],{digit}, [“.”], {digit}, [exp];**

**digit= “0” |“1” |“2” |“3” |“4” |“5” |“6” |“7” |“8” |“9” ;**

**sign= “+” | “-”;**

**exp= “E”, [sign] ,digit ,{digit}, [“.”], {digit};**

1. כעת כתבו regular expression המגדיר את אותו הליטרל. **ביטוי רגולרי ללא הסבר מספק לא יתקבל.**

**הערה:** מטרתכם היא לתפוס את כל הליטרל, ללא תלות בערך שהוא מייצג. לכן גם הליטרלים: 00.3400, 0.0-, 014, 3-, 7. , 1.3+ הם כולם ליטרלים חוקיים.

מומלץ להשתמש במנוע [www.regexr.com](http://www.regexr.com/)  לבדיקת נכונות הביטוי.

**([-]|[+])?[0-9]\*([.][0-9]+)?[0-9]\*([E](([-]|[+])?[0-9]+))?[0-9]\*([.][0-9]+)?[0-9]\***

1 אז בהתחלה שמנו את "+" או " -" כאופציה כי זה לא הכרחי =**([-]|[+])?**

2 לאחר מכן אפשרות לשים ספרות ככל שרוצים עם נקודה בינהם עם מספר אחד לפחות אחרי הנקודה **=[[0-9]\*([.][0-9]+)?[0-9]\***

3 אפשרות בחירה לשים E ואופציה לסימן אח"כ אבל חייב לשים ספרה אחת לפחות אחריהם=**([E](([-]|[+])?[0-9]+))?**

4 ושוב כמה ספרות שרוצים עם נקודה= **[0-9]\*([.][0-9]+)?[0-9]\***

שאלה 4:

ב-ML ניתן להגדיר פונקציה "בתוך" פונקציה אחרת, למשל:

fun foo () =  
    let

        fun bar x = 2 \* x

    in

        bar

    end;

דבר זה אפשרי גם ב-Rust, למשל:

fn foo() -> fn(i32)->i32 {

    fn bar(x: i32) -> i32 { 2 \* x }

    bar

}

1. מצא הבדל בין השפות בהקשר זה. (מה המגבלות על פונקציות פנימיות בכל שפה?)

בRust חייבים לציין את סוג הדברים המסוימים ולא קיימת לא מערכת הסקה אוטומטית כמו לML.

1. הבא קטעי קוד המדגימים את ההבדל.