שפות תכנות, 236319

אביב 2021

תרגיל בית 2

קארן לנדר- 345088199  
דור רוזן- 318965365

# **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 (object oriented)

כמו Java, Scala היא שפה מונחית עצמים במובן שכל ערך הוא אובייקט (אפילו integers וdoubles). Scala היא גם שפה פונקציונלית כי כל פונקציה היא ערך ואפשר להשתמש בה בכל מקום בו אתה יכול להשתמש בערך.

* **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 שמתאפיינת ביכולות ביצועיות גבוהות, שמה דגש על בטיחות ומהירות.

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

שפת מונחת עצמים טהורה. מכילה שילוב של ביטויים רגלוריים. יש לה איסוף זבל.

* **Javascript***:* [*functional*](https://en.wikipedia.org/wiki/Functional_programming)*, [imperative](https://en.wikipedia.org/wiki/Imperative_programming" \o "Imperative programming)*

שפת תכנות המותאמת לשילוב באתרי אינטרנט ורצה על ידי דפדפן האינטרנט בצד הלקוח.

השפה משתמשת בפירוש ולא בהידור (דומה לפייתון). השפה תומכת בתחביר סטנדרטי של שפת C אך היא גם סוג של מונחת עצמים. בנוסף השפה היא מונחת אירועים.

* **Clojure**: [concurrent](https://en.wikipedia.org/wiki/Concurrent_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming" \o "Functional programming)

דיאלקט מודרני של LISP שלמדנו בכיתה. השפה מעודדת תכנות פונקציונלי ומיושבת על JVM.

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

שפת שאילתות למסדי נתונים. הפקודות בשפה נועדו לאחזר ולטפל בנתונים.

* **ATS**:  [functional](https://en.wikipedia.org/wiki/Functional_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming" \o "Imperative programming)

שפת תכנות שנועדה לאחד תכנות עם מפרט רשמי. השפה היא פונקציונלית ואימפרטיבית. ניתן לראות את הגישה לתכנות אימפרטיבית ב- ATS בפרדיגמה של תכנות

theorem-proving. מערכת הטיפוסים של ATS מאפשרת תכונות רבות שנחשבות מסוכנות בשפות אחרות (למשל, חשבון מצביע מפורש והקצאת זיכרון) ,מה שהופך את ATS לשפת לתכנות מערכות ברמה נמוכה.

## שאלה 2:

tail-recursion (רקורסיית זנב)

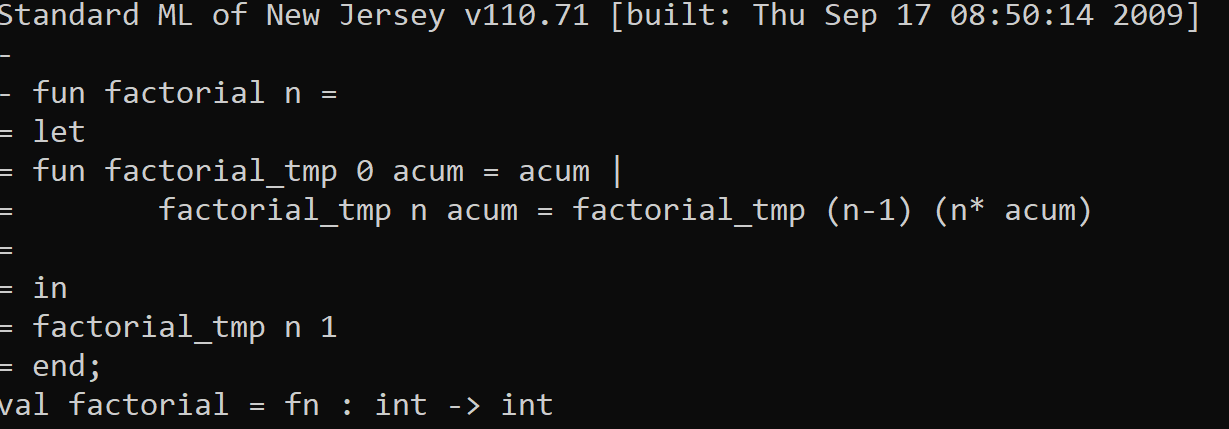
1. הסבר מה זה tail-recursion.

פונקציה רקורסיבית היא tail recursive כאשר הקריאה הרקורסיבית היא הדבר האחרון שמופיע בתוך הפונקציה ואין צורך לעלות בחזרה ברקורסיה כדי לבצע חישובים נוספים. רקורסיית זנב נחשבת כטובה יותר מרקורסיה רגילה מפני שניתן לבצע אופטימיזציות ע"י המהדר. למשל, אין צורך לשמור את מחסנית הסביבה של הפונקציה הנוכחית.

1. איך ניתן להשתמש ב-tail-recursion כדי ליעל תוכניות בשפות פונקציוליות?

בתוכניות בשפות פונקציונליות רקורסיה היא הדרך הנפוצה ביותר (ולפעמים הדרך היחידה כמו בML) לבצע איטרציות. לא לשמור על מחסנית הסביבה מאפשר להשתמש בכמות דומה של זיכרון כלולאה. לכן נרצה לשנות את הקוד הרקורסיבי כדי שיהיה זנב-רקורסיבי ונוכל לנצל את התכונה הזו. לעיתים קרובות הדבר דורש הוספת משתנה "acummulator" לפונקציה.

1. כתוב ב-ML פונקציה שהיא tail-recursive המקבלת מספר טבעי ומחזירה את .



## שאלה 3:

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

ניתן לתאר דקדוק של שפה ע"י EBNF וגם ע"י BNF. EBNF מהווה הרחבה לBNF וההבדל העיקרי הוא שגוף הביטוי יכול להיות גם ביטוי רגולרי מעל אוסף הסימנים. שימוש בביטויים רגולריים כאלו הוא Syntactic Sugar לדקדוקי BNF.

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.

ביטיו EBNF:

Expression = Integer | Float | Scientific;

Integer = [Sign], Number, {Number} ;

Float = Integer, “.” Number, {Number} ;

Scientific = Integer, "E", [Sign], Number, {Number} | Float, "E", [Sign], Number, {Number} ;

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

= Sign " ;"-"|"+

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.

## ביטוי רגולרי המגדיר את ליטרל המספרים:



## הסבר: נרצה שבהתחלה יופיע פעם אחת או כלום סימן "+" או "-". לאחר מכן נדרוש שיופיע אפס או יותר ספרות המייצגות את החלק השלם של המספר. לאחר מכן יופיע פעם אחת או אפס פעמים "." ונדרוש שיופיע לפחות ספרה אחת. בהמשך, אם מופיע "E" נרצה שיופיע פעם אחת או כלום סימן "+" או "-" ואז ולאחר מכן נדרוש שיופיע לפחות ספרה אחת.

נשים לב כי ביטוי כמו 1.5E אינו חוקי מכיוון שאחרי האות E חייב להופיע ספרה אחת.

## 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(y:i32) -> 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?)

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

1. הבא קטעי קוד המדגימים את ההבדל.