**עקרונות שפות תכנות - תרגיל 1**

**נינל בניוש 324699826**

**מורז תמיר 208397455**

**חלק תיאורטי**

**שאלה 1: מודלים לשערוך**

**סעיף א':**

הפעולה define אינה חלק מתכנות פונקציונלי טהור.

בתכנות פונקציונלי טהור אין השמות ולכן אין state.

פעולת define היא special form המקבל שם של משתנה וביטוי אחר, יש שמירת מצב – state. אין אפשרות לשערך בצורה רגילה, לדוגמא (define x 5) הכנסנו את 5 לתוך x.

אם define הייתה ממומשת על ידי פרוצדורה ומחזירה תמיד את הערך המוגדר אז הבעיות לא היו נפתרות, עדיין קיימות בעיות פונקציונליות, לדוגמה השימוש ב- define יכול לגרום לבעיות תחרות כאשר מספר תהליכים מנסים לגשת לקבועים משותפים.

לפרוצדורה ניתן שם, וניתן להגדיר פונקציה ע"י קיצור סינטקטי, אך define אינה פונקציה לכן אי אפשר להעביר אותה כארגומנט לפונקציה אחרת ולהשתמש בה כתוצא של פונקציה.

**סעיף ב':**

סמנטיקה היא משמעות, קשורה לתהליך השערוך ולפלטים האפשריים כלומר איזה ערכים קיימים בשפה. לכן סמנטיקה קשורה למבנה הביטוי שמגדירה אילו פלטים יכולים להתקבל מהביטוי.

**סעיף ג':**

תחביר מתייחס לחוקים, מה מותר לעשות ובאיזה סדר, סמנטיקה מתייחסת למשמעות הביטוי.

אליס המציאה אלגוריתם שערוך חדש, המקבל ומחזיר את אותם ביטויים, לכן הסמנטיקה תשאר זהה כי מדובר בערכים המוחזרים, בכך נשמור על משמעות הביטוי אך התחביר ישתנה, השיטה של החישוב והמניפולציה על הביטוי תשתנה כדי לראות את השיפור של אליס, והערך המוחזר הוא אותו ערך.

**סעיף ד':**

בתכנות שאינו תכנות אימפרטיבי אין צורך להשתמש ב- side effects.

השימוש ב- side effects מתייחס לביטויים שמשנים את הסביבה, ובתכנות שאינו אימפרטיבי אין שינויים במערכת ובכך אין שמירת מצב.

**סעיף ה':**

דוגמאות ל- side effects: [מקור](https://courses.cs.washington.edu/courses/cse341/05au/lectures/scheme-side-effects.html)

1. שימוש ב- current-time: יצירת קשר עם סביבת מערכת ההפעלה ולקיחת השעה הנוכחית.

כתיבה: (define current-time(current-time))

2. פונקציות I/O

read - קריאה מאובייקט בשפה והחזרתו, דוגמה: (define clam (read))

write – הדפסת ייצוג של הארגומנט.

display – דומה לכתיבה, אך משמיטה מירכאות סביב מחרוזת, הדפסה ללא מירכאות.

**סעיף ו':**

Prolog ממומש בתכנות לוגי, פקודה זו מאפשרת לכתוב סדרת עובדות, היא לא קשורה לתכנות אימפרטיבי או פונקציונאלי משום שהיא מבוססת על לוגיקה, כלומר מגדירים חוקים לוגים ואפשר לבדוק מה עומד בהם ומה לא.

ב- prolog לא נגדיר איזו בעיה התוכנה צריכה לפתור של אחרי שלב כמו בתכנות אימפרטיבי או פונקציונלי, אלא יש להגדיר את הבעיה בצורה לוגית.

**שאלה 2: SF - special forms**

**סעיף א':**

(define var\_name lambda)

בביטוי זה אנו מבצעים השמה של הערך lambda למשתנה var\_name, חוקי ב- scheme ואינו SF.

**סעיף ב':**

Define מבצע שינוי של הסביבה, ולכן הוא תמיד side-effect, הוא מבצע binding של שם המשתנה שהגדרנו עם הערך שחזר, לכן אין ביטוי חוקי בשפת scheme המשתמש ב- define ואינו SF.

**סעיף ג':**

הביטוי לא חייב להיות special form בשפת Scheme מאחר שפונקציות בשפה יכולות לקבל כארגומנטים גם ביטויים, בסדר כלשהו, ולכן אין משמעות לסדר ונוכל לשערך לפי applicative order, במידה והייתה דרישה לסדר השערוך ומשמעות לסדר אז special-form.

ניתן להעריך בכל סדר ולהחליט איזו תוצאה להחזיר בהתאם ל e3.

**סעיף ד':**

ביטוי if-reverse שעבורם יוחזר אותו ערך:

1. (reverse-if 10 3 true)
2. (reverse-if 6 (- 4 1) true)

כפי שצוין בסעיף ג', אם e3 הוא true נקבל את e2 ולכן התוצאה שנקבל בכל אחת מהדוגמאות תהיה 3.

**סעיף ה':**

הביטוי מבצע תחילה שערוך ל- e3 ובהתאם יוחזר e2 או e1.

ביטוי זה חייב להיות special form, מכיוון שהסדר שערוך שונה מהרגיל לא משמאל לימין.

התוצאה של הערך המוחזר תלויה בערך e3. יש שערוך מיוחד ולכן sf.

**שאלה 3: תהליך Evaluation**

**ביטוי 1:**

>> Error, you can’t divide zero by zero.

**ביטוי 2:**

>>

* Evaluate
  + Evaluate + → primitive procedure +
  + Evaluate 6 → Number 6
  + Evaluate 7 → Number 7
  + Apply + to (Num 6, Num 7)
  + Return 13 as result.
* Evaluate 13 → Number 13
* Evaluate ()
  + Return 13 as result.

>>

>>

>> (640)

**ביטוי 3:**

>>

>>

>>

* Bind <x,4>
* Evaluate e1:
  + #<procedure>
  + <closure (x) (- x 1)>→ <closure params: (x) body: ( (\* x y) )
* Evaluate e2: <4> return value 4.
* Apply value <e1> to value <e2>: x=4 → (- 4 1) → 3

>>

>> Return 36 as result

**ביטוי 4:**

>>

* Bind <a,7>, Bind <b,8>

>>

* Bind <x,8>, Bind <y,9>

>>

* #t→ eval <> → return 24.

**ביטוי 5:**

>>

* Evaluate e1: return <closure (a)…)>
* Evaluate + → primitive procedure +
* Bind <a,+>

>> ((lambda (a)…)(\* 4 4))

* Evaluate e1: return <closure (a)…)>
* Evaluate (\* 4 4) → return 16.
* Bind <a,16>

>>

* Evaluate (+ 6 -1 16)
  + Evaluate + → primitive procedure +
  + Evaluate 6 → Number 6
  + Evaluate -1 → Number -1
  + Evaluate 16 → Number 16
  + Apply + to (Num 6, Num -1, Num 16) → result 21
* Evaluate lambda (b)
  + Bind <b, 21>

>>

* Evaluate lambda (c)
  + Bind <c, 77>

>>→ return 119

**שאלה 4**

**סעיפים א'+ב:**

עבור כל ביטוי –טיפוס הביטוי, טיפוס הערך, צורה מיוחדת או ביטוי הפעלה.

ביטוי 1:

* טיפוס הביטוי: ביטוי פונקציה
* טיפוס הערך: פונקציה
* צורה מיוחדת: lambda, ביטוי המגדיר פונקציה המקבלת ארגומנט אחד x . גוף הפונקציה מכיל משוואת חילוק של 0 ב-0, פעולה לא חוקית.

ביטוי 2:

* טיפוס הביטוי: ביטוי הפעלה
* טיפוס הערך: מספר
* ביטוי הפעלה: מורכב מ-2 חלקים, תנאי if ופעולה 80/8. במידה והתנאי מתקיים נבצע את הפעולה.

ביטוי 3:

* טיפוס הביטוי: ביטוי הפעלה
* טיפוס הערך: מספר
* ביטוי הפעלה: מורכב מביטויי if ו- lambda, התנאי הפנימי תמיד מחזיר #t, הארגומנט הראשון הוא #f , בתנאי החיצוני ניקח את הפונקציה הראשונה כי התשובה שקיבלנו #t.

ב- lambda ביצענו השמה x=5 ולבסוף מתבצעת פעולה של הכפלה.

ביטוי 4:

* טיפוס הביטוי: ביטוי הפעלה, פרוצדורה
* טיפוס הערך: מספר
* ביטוי הפעלה: הביטוי מורכב, משלב רשימות, lambda וביטויים ארגומנטים.

ביטוי 5:

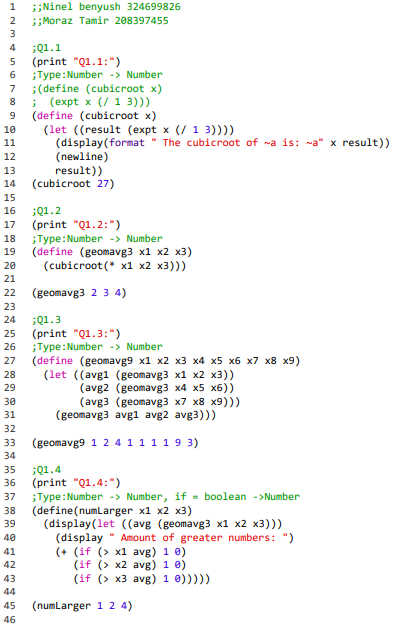
* טיפוס הביטוי: ביטוי הפעלה, פרוצדורה
* טיפוס הערך: מספר
* ביטוי הפעלה: הביטוי מורכב מקינון של פונקציות lambda .

**חלק תכנותי**

השתמשנו באתר - [jdoodle](https://www.jdoodle.com/execute-scheme-online) לכתיבת הקוד, עקב בעיה בהתקנת התוכנה של וירוסים, נציין שהוספנו פקודות הדפסה לכל סעיף, בעוד שב- drRacket אין צורך.

את הקוד נוסיף בכתבן, ונראה דוגמאות של פלט.

**שאלה 1:**



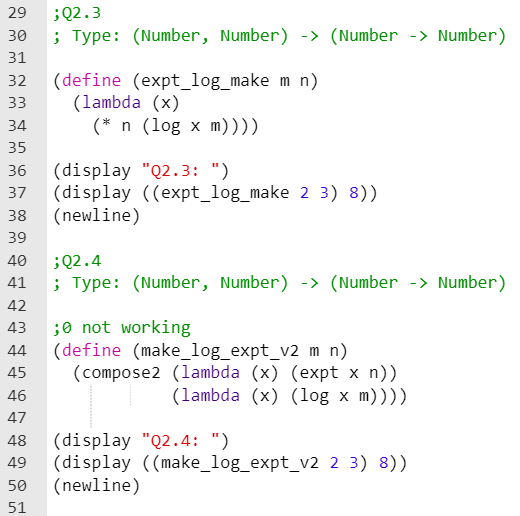
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated**שאלה 2:**

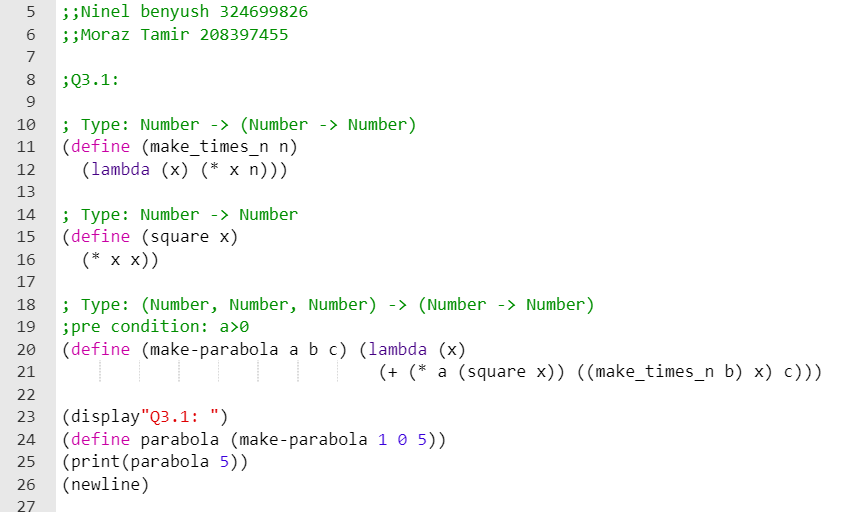
A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

**שאלה 3:**



A screenshot of a computer program

Description automatically generated

A screenshot of a phone

Description automatically generated A screenshot of a computer program

Description automatically generated