# מושגים בשפות תכנות, שנה"ל תשע"ו, סמסטר ב', מועד א'

**פרופ' מולי שגיב, עודד פדון**

**הנחיות כלליות:**

- משך הבחינה 3 שעות.
- מותר להשתמש בכל חומר כתוב (אין להשתמש במחשבון, מחשב, או פלאפון).
- בכל השאלות עליכם לתת הסבר משכנע מדוע התשובה שכתבתם נכונה, אלא אם מצויין אחרת.
- במבחן 7 שאלות. שאלה 1 היא חובה, ועליכם לבחור לענות על 5 שאלות מבין שאלות 2-7.
- את כל התשובות יש לכתוב במחברת הבחינה בלבד, תשובות על טופס הבחינה לא ייבדקו.
- חלוקת הניקוד: שאלה 1, 10 נקודות, שאלות 2-7, 18 נקודות כל אחת. סה"כ 100 נקודות.

נא להקיף בעיגול את השאלות שבחרת לענות עליהן:

1 2 3 4 5 6 7

**חובה לענות על השאלה הבאה:**

**שאלה 1 - שאלת חובה (10 נקודות)**

בשאלה זו עלייך לסמן נכון / לא נכון בכל אחד מהסעיפים (בשאלה זו אין צורך לנמק). תשובה נכונה מזכה ב 2 נקודות לכל סעיף, ותשובה לא נכונה לא משפיעה על הציון.

1. בדיקת הטיפוסים בשפת Scala מבוצעת בזמן ריצה.
2. בשפת Go, שימוש ב go-routines יכול לשפר ביצועים של תוכניות מקביליות.
3. ביצוע של Z-combinator בסמנטיקת lazy-evaluation גורם לחישוב אינסופי.
4. בשפת C אין צורך ב access link כדי לגשת למשתנים לא לוקאליים.
5. התוכניות הבאות שקולות תחת Structural Operational Semantics:

- `x := 2`
- `x := 1; x := x + 1`

**בחרי 5 שאלות מהשאלות הבאות. משקל כל שאלה 18 נקודות.**

**שאלה 2 - Parsing**

נתבונן בדקדוק הבא עבור ביטויים רגולריים:

```
R ::= c | R "." R | R "|" R | R "*" | "(" R ")"
```

כאשר R הוא non-terminal, וה terminals הם:

```
c, ".", "|", "*", "(", ")"
```

א. הראי שהדקדוק אינו חד משמעי ע"י מתן שני עצי גזירה שונים לאותה מילה.

ב. כתבי דקדוק חד משמעי שקול (עבור אותה שפה), כך שהאופרטור בעל הקדימות העליונה הוא כוכב (*), לאחריו שרשור (.) ולאחריו איחוד (|). שרשור ואיחוד הם בעלי אסוציאטיביות שמאלית. לדוגמה, המילה הבאה:

```
c.c.c*|c.c|c
```

מתפרשת כ:

```
(((c.c).(c*)) | (c.c)) | c
```

ג. ציירי עץ גזירה לפי הדקדוק שכתבת בסעיף ב', עבור המילה הבאה:

```
c.c.c*
```

ד. כתבי מהן קבוצות ה NULLABLE, FIRST, FOLLOW, SELECT עבור הדקדוק שכתבת בסעיף ב'.
האם הדקדוק הוא LL(1)? כיצד זה מתבטא בקבוצות הנ"ל?
הערה: אין צורך לכתוב את הקבוצות עבור terminals, אלא רק עבור non-terminals.

**שאלה 3 - סמנטיקה**

נרצה להוסיף לשפת While את הפקודה הבאה:

**while odd** $b_1$ **even** $b_2$ **do** S

זוהי לולאה עם שני תנאי עצירה, אחד שנבדק לפני איטרציות אי-זוגיות ($b_1$) ואחד שנבדק לפני איטרציות זוגיות ($b_2$). האיטרציה הראשונה היא אי-זוגית, ולכן התנאי שנבדק לפני הביצוע הראשון של הלולאה הוא $b_1$.

לדוגמה, עבור התוכנית הבאה:

**while odd** n < 4 **even** n < 3 **do** n := n+2

אם נריץ את התוכנית ממצב בו n=0 היא תסתיים במצב בו n=4 (התנאי $b_1$ יופר לפני האיטרציה השלישית).

אם נריץ את התוכנית ממצב בו n=1 היא תסתיים במצב בו n=3 (התנאי $b_2$ יופר לפני האיטרציה השניה).

א. הרחיבי את ה Natural Operational Semantics כדי לטפל בפקודה החדשה. הכללים אינם יכולים להסתמך על מבנה לולאת while הרגילה בשפה (כלומר לא ניתן להתייחס בכללים למבנה לולאת while הרגילה, אלא רק לפקודת הלולאה החדשה). הכללים חייבים להוות הגדרה אינדוקטיבית כפי שלמדנו בקורס, ואסור להם להוסיף משתנים נוספים לשפה.

ב. הדגימי את הסמנטיקה המורחבת שהגדרת בסעיף א' ע"י בניית עץ גזירה לתוכנית הדוגמה מפסקת הפתיחה ממצב התחלתי בו n=1.

ג. הרחיבי את ה Structural Operational Semantics כדי לטפל בפקודה החדשה. הכללים אינם יכולים להסתמך על מבנה לולאת while הרגילה בשפה (כלומר לא ניתן להתייחס בכללים למבנה לולאת while הרגילה, אלא רק לפקודת הלולאה החדשה). הכללים חייבים להוות הגדרה אינדוקטיבית כפי שלמדנו בקורס, ואסור להם להוסיף משתנים נוספים לשפה.

ד. הדגימי את הסמנטיקה המורחבת שהגדרת בסעיף ג' ע"י בניית סדרת גזירה לתוכנית הדוגמה מפסקת הפתיחה ממצב התחלתי בו n=0.

**שאלה 4 - $\lambda$ -calculus**

נתון הביטוי הבא ב $\lambda$ -calculus:

$(\lambda u.\ \lambda v.\ v\ v)\ ((\lambda s.\ \lambda z.\ s\ z)\ \lambda x.x)\ \lambda x.x$

א. ציירי את ה AST המתאים לביטוי.

ב. כתבי סדרת חישוב (reduction) לביטוי תחת call by value semantics.

ג. כתבי סדרת חישוב (reduction) לביטוי תחת lazy evaluation semantics.

ד. הסבירי בקצרה את ההבדל בין call by value ו lazy evaluation.

**שאלה 5 - OCaml**

הבהרה: בשאלה זו אסור להשתמש בפונקציות ספרייה מהמודול List

א. כתבי פונקציית OCaml בשם count_pairs. הפונקציה count_pairs מקבלת שני פרמטרים: הראשון הוא פונקציה שמחזירה ערך בוליאני, והשני הוא רשימה. count_pairs מחזירה את מספר הזוגות של איברים סמוכים ברשימה, שהפעלת הפונקציה המתקבלת עליהם מחזירה true. הפונקציה count_pairs תמיד תחזיר 0 אם הרשימה מכילה פחות משני איברים.

הנה מספר דוגמאות לשימוש ב count_pairs:

```
# count_pairs (fun x y -> x < y) [1; 2; 3; 4; 5];;
- : int = 4
# count_pairs (fun x y -> x < y) [1; 2; 1; 7; 5];;
- : int = 2
# count_pairs (=) ["Python"; "OCaml"; "JavaScript"; "Scala"; "Python"];;
- : int = 0
# count_pairs (=) ["OCaml"; "OCaml"; "JavaScript"; "JavaScript"];;
- : int = 2
```

ב. מהו הטיפוס הכללי ביותר של הפונקציה שכתבת בסעיף א'?

ג. מהי צריכת הזיכרון של הפונקציה שכתבת בסעיף א' כתלות באורך הרשימה שהיא מקבלת? הסבירי היכן זיכרון זה מוקצה, והאם ניתן להפעיל את הפונקציה על רשימה של מאה מיליון איברים?

ד. אם צריכת הזיכרון של הפונקציה שכתבת בסעיף א' אינה קבועה, כתבי גרסה שלה שיכולה לפעול בזיכרון קבוע ללא תלות באורך הרשימה (רמז: רקורסיית זנב). מותר להגדיר פונקציית עזר.

**שאלה 6 - JavaScript**

כחלק ממימוש של שרת הודעות ב JavaScript, אנו רוצים לאפשר למשתמש לבקש מהשרת לשלוח הודעה עם השהייה. כחלק ממימוש השרת, כתבנו לצורך כך את הפונקציה send_after_delay, שמקבלת הודעה, ומספר שקובע כמה שניות לחכות לפני שליחתה. כמו כן, לאובייקט השרת יש מתודה now שמחזירה את הזמן הנוכחי בשניות, וכן מתודה send ששולחת הודעה באופן מיידי. להלן הקוד שכתבנו לפונקציה send_after_delay:

```
function send_after_delay(msg, delay) {
   var t = this.now(); // returns current time in seconds
   while (this.now() < t + delay) {} // wait for delay seconds
   this.send(msg);
}
```

א. הסבירי בקצרה מה הבעיה במימוש הנ"ל, בהנחה שיש משתמשים רבים שניגשים לשרת, וכולם רוצים לשלוח הודעות מושהות.

ב. תקני את הקוד ע"י שימוש ב closure ובפונקציית ספריה מתאימה (אם אינך זוכרת את פונקציית הספריה המתאימה, המציאי עבורה שם ותארי במדויק מה היא עושה).

ג. הסבירי איך המימוש המתוקן שלך מסעיף ב' מתמודד בהצלחה עם משתמשים רבים שניגשים לשרת ושולחים הודעות מושהות.

**שאלה 7 - Types**

נתון המימוש הבא של פונקציית OCaml שאמורה לבדוק האם איבר מופיע ברשימה:

```
let rec present x xs = match xs with
 | [] -> false
 | h::t -> present x t
```

א. נתחי את הטיפוס הכללי ביותר של הפונקציה present.

ב. כיצד ניתן להבין מהטיפוס שיש באג במימוש?

ג. תקני את הבאג.

ד. נתחי את הטיפוס של הפונקציה המתוקנת.

**בהצלחה,**

**מולי ועודד**