בוחן בקורס: עקרונות שפות תכנות, 202-1-2051
תאריך: 2/5/2019
שמות המרצים: מני אדלר, בן אייל, גיל אינציגר, מיכאל אלחדד, ירון גונן
מיועד לתלמידי: מדעי המחשב והנדסת תוכנה, שנה ב', סמסטר ב'
משך הבוחן: שעתיים
חומר עזר: אסור

סה"כ 105 נקודות

הנחיות כלליות:

- מבחנים שייכתבו בעיפרון חלש, המקשה על הקריאה, לא ייבדקו.
- יש לענות על כל השאלות בגיליון התשובות. מומלץ לא לחרוג מהמקום המוקצה.
- אם אינכם יודעים את התשובה, ניתן לכתוב 'לא יודע' ולקבל 20% מהניקוד על הסעיף/השאלה.

בהצלחה!

שאלה 1 [25 נקודות]

א. [9 נקודות]

ממשו ב-TypeScript את הפונקציה curry, מקבלת פונקציה של שני פרמטרים ומחזירה פונקציה של פרמטר אחד שמחזירה פונקציה של פרמטר אחד. לדוגמא:

```
const add: (x: number, y: number) => number = (x, y) => x + y;
const curriedAdd = curry(add);
console.log(curriedAdd(1)(5)); // => 6
```

המימוש שלכם צריך להיות גנרי. יש לציין את הטיפוסים.

ב. [9 נקודות]

ניתנות הגדרות של טיפוסים לפי שיטת disjoint-union:

```
type Shape = Circle | Rectangle;
interface Circle { tag: "circle"; center: Point2D; radius: number; }
interface Rectangle { tag: "rectangle"; upperLeft: Point2D; lowerRight:
Point2D; }
```

הגדירו את ה-type predicates. הקפידו לציין טיפוסים:

```
const isShape =

const isCircle =

const isRectangle =
```

ג. [7 נקודות]

ציירו את ה-AST של הביטוי הבא:

```
((if (< x y) - +) (+ 1 2) 3)
```

שאלה 2: תכנות פונקציונלי [30 נקודות]

בשאלה זו ניתן להשתמש רק בפרימיטיבים של L3, אלא אם כן צוין אחרת.

א. [10 נקודות]

ממשו ב-L3 את הפרוצדורה `reverse`, המקבלת רשימה ומחזירה רשימה עם אותם איברים בסדר הפוך.
לדוגמא:

```
(reverse '(1 2 3)) → '(3 2 1)
(reverse '()) → '()
```

ניתן להשתמש ב-`append` (שמומש כבר בתרגיל 2).
השלימו את החלקים החסרים בחוזה (contract) של הפרוצדורה בגיליון התשובות.

ב. [10 נקודות]

ממשו ב-L3 את הפרוצדורה `map2`, המקבלת פונקציה של 2 פרמטרים ושתי רשימות בעלות אותו אורך, ומחזירה רשימה באותו אורך, כאשר האיבר ה-i ברשימה המוחזרת הוא תוצאת הפעלת הפונקציה על האיבר ה-i בשתי הרשימות. לדוגמא:

```
(map2 + '(1 2 3) '(4 5 6)) → '(5 7 9)
```

השלימו את החלקים החסרים בחוזה (contract) של הפרוצדורה בגיליון התשובות.

ג. [10 נקודות]

ממשו ב-L3 את הפונקציה `derive` המקבלת פונקציה f ומידת קירוב dx, ומחזירה פונקציה המחשבת קירוב לנגזרת שלה על פי הנוסחה:

```
f'(x) = [f(x+dx)-f(x)] / dx
```

לדוגמא:

```
(define square (lambda (x) (* x x)))
(define f1 (derive square 0.001))
(define f2 (derive square 0.1))

(f1 2) → 4.000999999999699 (Real value is 4)
(f2 2) →  4.100000000000001
```

השלימו את החלקים החסרים בחוזה (contract) של הפרוצדורה בגיליון התשובות.

שאלה 3: תרגום קוד [30 נקודות]

א. [12 נקודות]

א.1 [6 נקודות]

נתונות שתי שפות sourceL, targetL, המבוססות על השפות L1-L3 (כמו לדוגמא L2 L3, או L20 L30 בתרגיל 2).

בתחביר המופשט (AST) של השפה sourceL יש מבנה תחבירי שאינו קיים בתחביר המופשט של target.

- תנו דוגמא לשתי שפות כאלה (ניתן להמציא שפות כרצונכם).
- ציינו איזה תנאי מאפשר לתרגם AST של תוכנית ב sourceL לתוכנית שקולה סמנטית בשפה targetL.
- תנו דוגמא לקיום תנאי שכזה.

א.2 [6 נקודות]

נתונות שתי שפות: sourceL, targetL, המבוססות על השפות L1-L3 (כמו לדוגמא L2 L3, או L20 L30 בתרגיל 2).

לשתי השפות יש אותו תחביר מופשט (AST), אך באינטרפרטר של sourceL קיים אופרטור פרימיטיבי שאינו קיים באינטרפרטר של targetL.

- תנו דוגמא לשתי שפות כאלה (ניתן להמציא שפות כרצונכם).
- ציינו איזה תנאי מאפשר לתרגם AST של תוכנית ב sourceL לתוכנית שקולה סמנטית בשפה targetL.
- תנו דוגמא לקיום תנאי שכזה.

ב. [18 נקודות]

להלן קוד פתרון שאלה 4 בעבודה 2 (כפי שפורסם), לתרגום AST של תוכנית ב L2 לתוכנית שקולה סמנטית ב Python:

```
/*
Purpose: Transform L2 AST to Python program string
Signature: l2ToPython(l2AST)
Type: [Parsed | Error] => [string | Error]
*/
export const l2ToPython = (exp: Parsed | Error): string | Error =>
    isError(exp) ? exp.message :
    isProgram(exp) ? map(l2ToPython,exp.exps).join("\n") :
    isBoolExp(exp) ? (exp.val ? 'True' : 'False') :
    isNumExp(exp) ? exp.val.toString() :
    isVarRef(exp) ? exp.var :
    isDefineExp(exp) ? exp.var.var + " = " + l2ToPython(exp.val) :
    isProcExp(exp) ? "(" + "lambda " +
                         map((p) => p.var, exp.args).join(",") + ": " +
                         l2ToPython(exp.body[exp.body.length-1]) +
```

```
                    ")" :
    isIfExp(exp) ? "(" + l2ToPython(exp.then) +
                        " if " +
                        l2ToPython(exp.test) +
                        " else " +
                        l2ToPython(exp.alt) +
                  ")" :
    isAppExp(exp) ?
        (isPrimOp(exp.rator) ?
            primOpApp2Python(exp.rator, exp.rands) :
            l2ToPython(exp.rator) + "(" +
                       map(l2ToPython, exp.rands).join(",") + ")") :
    Error("Unknown expression: " + exp.tag);


const primOpApp2Python = (rator : PrimOp, rands : CExp[]) : string =>
    rator.op === "not" ? "(not " + l2ToPython(rands[0]) + ")" :
    rator.op === "and" ? "(" + map(l2ToPython,rands).join(" && ") + ")" :
    rator.op === "or" ? "(" + map(l2ToPython,rands).join(" || ") + ")" :
    "(" + map(l2ToPython,rands).join(" " +
                       (rator.op === '=' ? '==' : rator.op) + " ") + ")"
```

ב.1 [5 נקודות]

במידה וה AST הניתן כפרמטר לפונקציה `l2ToPython` מכיל כמה ביטויים ב `body` של `ProcExp`, האם תוכנית ה Python שתתקבל תהיה שקולה? נמקו

כזכור (בתרגיל 2), בשפה L30 נוספו על L2 ארבעה אופרטורים פרימיטיביים - cons, pair?, car, cdr - וכן literal expression <u>אחד</u> המייצג רשימה ריקה ()'.

ב.2 [3 נקודות]

תרגמו את הקוד הבא ב L30 ל Python, כאשר pairs ב L30 מתורגמים ל tuple של שני איברים ב Python (ראו חומר העזר בהמשך).

```
(define f
   (lambda (p)
      (+ (car p) (cdr p)))

(f (cons 1 2))
```

ב.3 [10 נקודות]

השלימו את הקוד בגיליון התשובות, כך שהפרוצדורה `l30ToPython`, המקבלת AST של תוכנית בשפה L30, תחזיר מחרוזת של תוכנית שקולה ב Python (ראו חומר העזר בהמשך).

<u>חומר עזר</u>:

- ניתן להגדיר ב Python מערך בגודל קבוע (tuple) על ידי סוגריים עגולים. לדוגמא:
  מערך ריק: ()
  מערך בגודל קבוע 2 עם המספרים 4,5: (4,5)
  מערך בגודל קבוע 2 עם מערך מקונן: ((3 ,2) ,1)

- ניתן לשלוף איבר ממערך ב Python ע"י ציון האינדקס שלו בסוגרים מרובעים. לדוגמא:

(1,2)[0]
→ 1
(1,2)[1]
→ 2

- ניתן לבדוק ב Python האם ביטוי הוא מערך בגודל קבוע בעזרת הפקודה isinstance:

```
isinstance(((), tuple)
→ True
isinstance((4, 5), tuple)
→ True
isinstance(2, tuple)
→ False
```

שאלה 4: אינטרפרטר של מודל ההצבה/ההחלפה [20 נקודות]

א. Applicative Order מול Normal Order

א.1 מה ההבדל בין applicative order לבין normal order ?
[4 נקודות]

א.2 הדגימו על־ידי תוכנית ב־L3 חישוב שמסתיים ב־Normal Order אבל שאינו מסתיים ב־Applicative Order
[4 נקודות]

א.3 תנו דוגמא אחת בה חישוב ב־Normal Order עדיף, דוגמא אחת בה חישוב ב־Applicative Order עדיף.
נמקו. [4 נקודות]

ב. ערכים וביטויים ליטרליים

ב.1 מדוע אנו נזקקים לפונקציה `valueToLitExp` (בפונקציה `applyProc` ב L3-eval) במימוש מודל ההצבה/ההחלפה (substitution model)?
[4 נקודות]

ב.2 מדוע אין צורך בפרוצדורה `valueToLitExp` ב normal-eval?
[4 נקודות]