

מבחן בקורס: עקרונות שפות תכנות, 2021-2022

מועד: ב

תאריך: 12/8/2024

שמות המרצים: מני אדלר, ירון גונן, יובל פינטר

מיועד לתלמידי: מדעי המחשב והנדסת תוכנה

משך המבחן: שעותיים

חומר עזר: אסור

הנחיות כלליות:

- יש לענות על כל השאלות בגיליון התשובות. מומלץ לא לחרוג מן המקום המוקצה.

- אם אינכם יודעים את התשובה, ניתן לכתוב 'לא יודע' ולקבל 20% מהניקוד על הסעיף/השאלה.

שאלה 1: תחביר וסמנטיקה \_\_\_\_\_ נק 35

שאלה 2: מערכת טיפוסים \_\_\_\_\_ נק 20

שאלה 3: תכנות פונקציונאלי, CPS, רשימות עצלות \_\_\_\_\_ נק 30

שאלה 4: תכנות לוגי \_\_\_\_\_ נק 15

סה"כ \_\_\_\_\_ נק 100

בהצלחה!

-----

## שאלה 1: תחביר וסמנטיקה [35 נקודות]

א. [5 נק'] כמה פריימים, מלבד הסביבה הגלובלית, נוצרים בביצוע תוכנית ב-L3, במודל ההצבה ובמודל הסביבות?

ב. [5 נק'] כמה פריימים, מלבד הסביבה הגלובלית, נוצרים בביצוע תוכנית ב-L3 המורחבת, הכוללת גם את מבנה ה-class, במודל הסביבות?

בסוף את תשובתכם על המימוש הבא של מבנה ה-class במודל הסביבות.

```
export type ClassExp = {tag: "ClassExp"; fields: VarDecl[]; methods: Binding[]; }
export type ClassValue = {
  env : Env,
  tag: "Class";
  fields: string[];
  methodNames: string[];
  methodProcs : CExp[];
}
export type ObjectValue = {
  env : Env,
  tag: "Object";
  methodNames: string[];
  methodProcs : Closure[];
}

const applicativeEval = (exp: CExp, env: Env): Result<Value> =>
  isNumExp(exp) ? makeOk(exp.val) :
  isBoolExp(exp) ? makeOk(exp.val) :
  isStrExp(exp) ? makeOk(exp.val) :
  isPrimOp(exp) ? makeOk(exp) :
  isVarRef(exp) ? applyEnv(env, exp.var) :
  isLitExp(exp) ? makeOk(exp.val) :
  isIfExp(exp) ? evalIf(exp, env) :
  isProcExp(exp) ? evalProc(exp, env) :
  isLetExp(exp) ? evalLet(exp, env) :
  isClassExp(exp) ? evalClass(exp,env) :
  isAppExp(exp) ? bind(applicativeEval(exp.rator, env),
    (proc: Value) =>
      bind(mapResult((rand: CExp) =>
        applicativeEval(rand, env), exp.rands),
```

```

        (args: Value[]) =>
            applyProcedure(proc, args))) :
makeFailure("let not supported (yet)");

const applyProcedure = (proc: Value, args: Value[]): Result<Value> =>
    isPrimOp(proc) ? applyPrimitive(proc, args) :
    isClosure(proc) ? applyClosure(proc, args) :
    isClassValue(proc) ? applyClass(proc, args) :
    isObjectValue(proc) ? applyObject(proc, args) :
    makeFailure(`Bad procedure ${format(proc)}`);

const applyClosure = (proc: Closure, args: Value[]): Result<Value> => {
    const vars = map((v: VarDecl) => v.var, proc.params);
    return evalSequence(proc.body, makeExtEnv(vars, args, proc.env));
}

const evalClass = (exp: ClassExp, env: Env): Result<ClassValue> =>
    makeOk(makeClassValue(map(vd=>vd.var, exp.fields), map(b=>b.var.var,
exp.methods), map(b=>b.val, exp.methods), env));

const applyClass = (cls: ClassValue, args: Value[]): Result<ObjectValue> => {
    const extEnv = makeExtEnv(cls.fields, args, cls.env);
    return bind(mapResult((method: CExp) =>
        applicativeEval(method, extEnv), cls.methodProcs),
        (methods: Value[]) =>
            allT(isClosure, methods) ?
                makeOk(makeObjectValue(cls.methodNames, methods, extEnv)) :
                makeFailure("Non-proc method"));
}

const applyObject = (object: ObjectValue, args: Value[]): Result<Value> => {
    if (args.length < 1)
        return makeFailure("Wrong number of parameters");
    else {
        if (isSymbolSExp(args[0]))
        {
            const i = object.methodNames.indexOf(args[0].val);
            if (i >= 0) {
                const proc : Closure = object.methodProcs[i];
                if (proc.params.length != args.length - 1)

```

```

        return makeFailure("Wrong number of parameters");
    return applyClosure(proc,args.slice(1));
} else return makeFailure(`Unrecognized method: ${args[0].val}`);
} else
    return makeFailure("Wrong type of first argument");
}
}

```

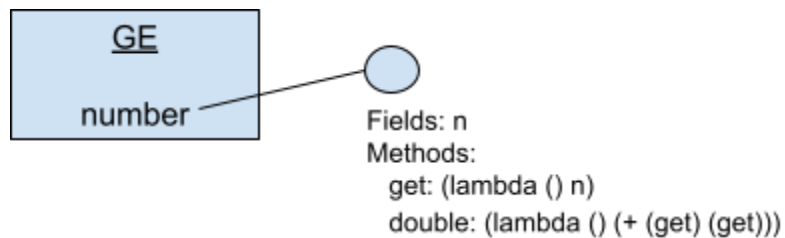
ג. [10 נק'] נצייר את דיאגרמת הסביבות עבור מחלקות ואובייקטים באופן הבא:

מחלקה:

```

(define number
  (class (n)
    ((get (lambda () n))
     (double (lambda () (+ (get) (get)))))
  )
)

```

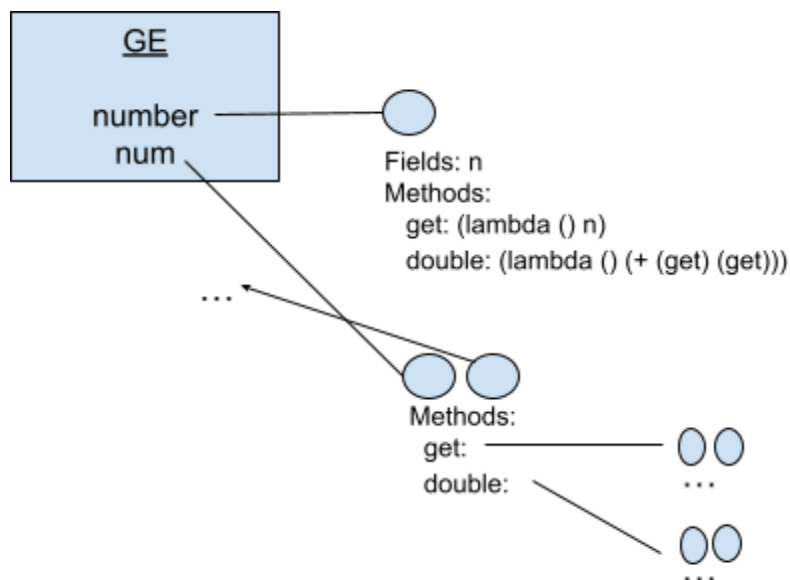


אובייקט:

```

(define num (number 5))

```



כאשר יש להחליף את שלוש הנקודות מתחת לקלוז'רים של המתודות `get`, `double` של `number` בתיאור הרגיל של קלוז'רים במודל הסביבות, ויש להחליף את שלוש הנקודות בסוף החץ שיוצא מהעיגול השני בתיאור האובייקט `num` לסביבה שבה הוא מוגדר.

השלימו את דיאגרמת הסביבות עבור חישוב התוכנית הבאה, והסבירו בעזרתה מה הבעיה בחישוב.

```
(define number
  (class (n)
    ((get (lambda () n))
     (double (lambda () (+ (get) (get)))))
  )
)
(define num (number 5))
(num 'double)
```

ד. [5 נק'] עדכנו את מימוש המתודה הבעייתית כדי שהבעיה הספציפית בסעיף ג לא תיווצר (לא נדרש פיתרון מערכתי אלא רק שה-`body` של המתודה יחושב ללא בעיה)

ה. [10 נק'] נתון כי בקובץ הסביבה של האינטרפרטר נוספה סביבה/פריים מסוג נוסף RecEnv:

```
export type Env = EmptyEnv | ExtEnv | RecEnv;
export type EmptyEnv = {tag: "EmptyEnv" }
export type ExtEnv = {
  tag: "ExtEnv";
  vars: string[];
  vals: Value[];
  nextEnv: Env;
}
export type RecEnv = {
  tag: "RecEnv";
  vars: string[];
  vals : ProcExp[];
  nextEnv: Env;
}

export const makeEmptyEnv = (): EmptyEnv => ({tag: "EmptyEnv"});
export const makeExtEnv = (vs: string[], vals: Value[], env: Env): ExtEnv =>
  ({tag: "ExtEnv", vars: vs, vals: vals, nextEnv: env});
export const makeRecEnv = (vs: string[], procs : ProcExp[], env: Env): RecEnv =>
  ({tag: "RecEnv", vars: vs, vals : procs, nextEnv: env});

export const applyEnv = (env: Env, v: string): Result<Value> =>
  isEmptyEnv(env) ? makeFailure(`var not found ${format(v)}`) :
  isExtEnv(env) ? applyExtEnv(env, v) :
  applyRecEnv(env, v);

const applyExtEnv = (env: ExtEnv, v: string): Result<Value> =>
  env.vars.includes(v) ? makeOk(env.vals[env.vars.indexOf(v)]) :
  applyEnv(env.nextEnv, v);

const applyRecEnv = (env: RecEnv, v: string): Result<Value> =>
  env.vars.includes(v) ? makeOk(makeClosure(env.vals[env.vars.indexOf(v)].args,
                                          env.vals[env.vars.indexOf(v)].body,
                                          env)) :
  applyEnv(env.nextEnv, v);
```

עדכנו את קוד האינטרפרטר שהופיע בסעיף ב כך שהבעיה בסעיף ג לא תיווצר.

## שאלה 2: טיפוסים [20 נקודות]

א. [12 נק'] האם הצהרות הטיפוס הבאות נכונות או לא? אין צורך בהסבר.  
ניתן להניח שבשפה הרלוונטית איחוד טיפוסים מוגדר, ומסומן בקו אנכי |.

1.  $\{f : [T2 \rightarrow T3], g : [T1 \rightarrow T4], a : \text{Number}\} \vdash (f (g a)) : T3$
2.  $\{a: \text{string}, b: \text{boolean}, x: \text{number}\} \vdash (\text{if } (< x 0) a b) : \text{string} \mid \text{boolean}$
3.  $\{f : [\text{number} * [T5 \rightarrow \text{number}] \rightarrow T6], y : T5\} \vdash (f (g y)) : T6$
4.  $\{a: \text{number}, b: \text{number}\} \vdash (+ a b) : \text{number}$

ב. [8 נק'] בתרגיל 3 עסקנו בייצוג של איחודים וחיתוכים בין טיפוסים. ראינו בין היתר כי נדרש להמיר את תוצאת האיחוד או החיתוך של שני ביטויי טיפוס לצורה הקנונית DNF (איחוד של חיתוכים). הוכיחו את הטענה הבאה, או הראו דוגמה נגדית: כל פעולת איחוד או חיתוך בין שני ביטויים הנתונים בצורת DNF ניתנת לחישוב והצגה כביטוי יחיד בצורת DNF ללא צורך בקריאה רקורסיבית. ניתן להניח כי מלבד איחודים וחיתוכים, השפה מורכבת מביטויי טיפוס אטומיים בלבד.

### שאלה 3: תכנות פונקציונאלי, CPS, רשימות עצלות [30 נקודות]

א. [8 נק'] כתבו את הפונקציה `deep-map` ב-L4, אשר מקבלת פונקציה, ועץ, המיוצג על-ידי רשימה מקוננת של רשימות, ומפעילה את הפונקציה על כל איבר בעץ, תוך שמירה על מבנה העץ. הפונקציה מחזירה את העץ החדש.

```
;; Signature:
;; [[T1 -> T2] * Tree<T1> -> Tree<T2>]
;; Example:
;; (deep-map add1 '((1 2) (3 (4 5) 6))) => '((2 3) (4 (5 6) 7))
(define deep-map
  (λ (f tree)
```

ב. [6 נק'] ציינו שני יתרונות לכך שה-'משתנים' בתכנות פונקציונאלי הם `immutable` (כלומר לא ברי-שינוי).

ג. [6 נק'] מה ההבדל בין תהליך החישוב של רקורסיית ראש ותהליך החישוב של רקורסיה הכתובה בסגנון CPS?

ד. [10 נק'] ממשו את הפונקציה `take-every-n-lzl` המקבלת רשימה עצלה ומספר `n`, ומחזירה רשימה עצלה עם כל איבר `n` של הרשימה המקורית:

```
;; Signature:
;; [Lzl<T1> * Number -> Lzl<T1>]
;; Example:
;; (take (take-every-n-lzl (ints-from 1) 7) 3) => '(7 14 21)
(define take-every-n-lzl
  (λ (lzl n)
    (letrec ((loop (λ (lzl counter)

      [...]

      (loop lzl n))))))
```



## שאלה 4: תכנות לוגי [15 נקודות]

א. [5 נק'] תארו את שלבי אלגוריתם היוניפיקציה עבור:

```
unify[ p([[v|V]|[V|VV]]), p([v|VV])]
```

ב. [10 נק'] ציירו את מסלולי ההצלחה בעץ ההוכחה עבור השאילתא הבאה (אין צורך לצייר מסלולים שלא מובילים לפיתרון)

```
parent(abraham, isaac).      %1
parent(abraham, ishmael).    %2
parent(ishmael, nevayot).     %3

ancestor(A, D):- parent(A,D).  %4
ancestor(A, D):- parent(P, D), ancestor(A, P).  %5

?-ancestor(A, nevayot)
```