دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

طراحی زبان‌های برنامه‌سازی

بهار ۱۴۰۱

استاد: محمد ایزدی
گردآورندگان: علی عباسی، سروش زارع

تمرین سوم **Data Abstraction** مهلت ارسال: ۱۱ اردی‌بهشت ۱۴۰۱

به موارد زیر توجه کنید:
۱) برنامه‌های خود را به زبان Racket بنویسید.
۲) مهلت ارسال تمرین ساعت ۵۹ : ۲۳ روز یک‌شنبه ۱۱ اردی‌بهشت ۱۴۰۱ است.
۳) در مسائلی که نیاز به خروجی دادن است، پاسخ را چاپ نکنید بلکه آن را به عنوان حاصل تابع برگردانید.
۴) حتما تابع اصلی خود را در main بگذارید تا هنگام داوری به مشکل نخورید. منظور از تابع اصلی تابعی است که اجرای آن روی ورودی، خروجی مطلوب را نشان می‌دهد.
۵) جواب هر سوال برنامه نویسی را در یک فایل Racket با نام و فرمت زیر قرار دهید.

$$[StudentID]\_[ProblemNumber].rkt$$

۶) در نهایت تمام فایل‌های Racket را در یک فایل زیپ قرار داده و با نام $HW3\_StudentID$ در سامانه کوئرا آپلود کنید.
۷) هرگونه سوالی راجع به تمرین را در زیر پست مربوطه در کوئرای درس مطرح کنید.
۸) در مجموع تمامی تمارین ۷ روز مهلت تاخیر مجاز دارید و پس از تمام شدن این تاخیرهای مجاز به ازای هر روز ۱۰ درصد از کل نمره تمرین شما کم می‌شود.
۹) لطفا تمرین‌ها را از یکدیگر کپی نکنید. در صورت وقوع چنین مواردی مطابق با سیاست درس رفتار می‌شود.

سوالات (۱۰۰ نمره)

۱. (۱۰ نمره) نمایش procedural برای environment را گسترش دهید و `empty-env?` را پیاده سازی کنید که تعیین می‌کند environment خالی است یا نه. برای این کار environment را به صورت لیستی از دو procedure نشان دهید که یکی از آن‌ها value متناظر با یک variable را بر می‌گرداند و دیگری تعیین می‌کند که environment خالی است یا نه.

۲. (۲۰ نمره) مجموعهٔ درختانی که با گرامر زیر توصیف می‌شوند را در نظر بگیرید:

Red-blue-tree ::= (`red-node` *Red-blue-tree Red-blue-tree*)
::= (`blue-node` {*Red-blue-tree*}*)
::= (`leaf-node` *Int*)

این نوع داده را به کمک دستور `define-datatype` تعریف کنید و به کمک آن تابعی بنویسید که یک درخت ورودی بگیرد و درختی با همان شکل خروجی دهد که اعداد برگ‌های آن، با تعداد گره‌های قرمز در مسیر ریشه به آن برگ جایگزین شده است.

۳. (۳۰ نمره) در این سوال می‌خواهیم توابعی برای ارزیابی عبارات ریاضی پیشوندی (Prefix Expression) طراحی کنیم. گرامر زیر را در نظر بگیرید:

Prefix-exp ::= *Int*
::= $(+|-|*)$ *Prefix-exp Prefix-exp*

به کمک دستور define-datatype، سینتکس انتزاعی زیر را برای آن تعریف می‌کنیم:

```
(define-datatype prefix-exp prefix-exp?
    (const-exp
        (num integer?))
    (op-exp
        (operator symbol?)
        (operand1 prefix-exp?)
        (operand2 prefix-exp?)))
```

(آ) تابع parse-prefix-list را به گونه‌ای بنویسید که با دریافت یک عبارت پیشوندی (به صورت لیست)، شئ prefix-exp مربوط به آن را تولید کند.

(ب) تابع main را به گونه‌ای بنویسید که با دریافت یک عبارت پیشوندی و با کمک تابع بخش قبل، حاصل آن عبارت را محاسبه کرده و به صورت یک عدد خروجی دهد.

فراخوانی نمونه:

```
(parse-prefix-list '(- - 3 2 + 4 - 12 7))
```

خروجی نمونه:

```
(op-exp
    '-
    (op-exp
        '-
        (const-exp 3)
        (const-exp 2))
    (op-exp
        '+
        (const-exp 4)
        (op-exp
            '-
            (const-exp 12)
            (const-exp 7))))
```

فراخوانی نمونه:

```
(main '(- - 3 2 + 4 - 12 7))
```

خروجی نمونه:

```
-8
```

۴. (۴۰ نمره) یک سیار دودویی شامل دو شاخه‌ی چپ و راست است. هر شاخه یک میله با یک طول است که از آن یک وزن یا یک سیار دودویی دیگر آویزان شده است. می‌توانیم سیار دودویی را به این صورت نشان دهیم:

```
(define (make-mobile left right)
    (list left right))
```

یک شاخه از ترکیب کردن یک طول (که باید عدد باشد) به همراه یک structure دیگر که می‌تواند یا یک وزن باشد یا یک سیار دودویی به وجود می‌آید:

```
(define (make-branch length structure)
    (list length structure))
```

(آ) دو تابع left-branch و right-branch را بنویسید به طوری که با دریافت یک سیار دودویی، شاخه‌های متناظر آن را برمی‌گردانند. همچنین دو تابع branch-length و branch-structure را بنویسید به طوری که به ترتیب طول یک شاخه و structure که از آن آویزان شده است (یا یک وزنه، یا یک سیار دودویی) را بر می‌گردانند. به این نوع تابع‌ها در اصطلاح selector گفته می‌شود.

(ب) با استفاده از selector هایی که نوشته‌اید، یک تابع total-weight بنویسید که مجموع تمام وزن‌های سیار را حساب می‌کند.

(ج) یک سیار دودویی را متعادل می‌نامیم هرگاه گشتاور ناشی از شاخه‌ی سمت چپ و راست آن با هم برابر باشن (در واقع طول شاخه‌ای که از سمت چپ آویزان شده است ضربدر وزن زیرسیار سمت چپ برابر باشد با طول شاخه‌ای که از سمت راست آویزان شده‌است ضربدر وزن زیرسیار سمت راست). یک تابع is-balanced? طراحی کنید که بررسی می‌کند یک سیار دودویی متعادل است یا خیر. به چنین تابع‌هایی در اصطلاح predicate می‌گویند.

(د) فرض کنید نمایش سیار دودویی را به این صورت تغییر داده‌ایم:

```
(define (make-mobile left right) (cons left right))
(define (make-branch length structure)
    (cons length structure))
```

به چه میزان باید توابع پیاده سازی شده خود را تغییر دهید به طوری که بر اساس این نمایش کار کنند؟