دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

طراحی زبان‌های برنامه‌سازی

دکتر محمد ایزدی

بهار ۱۴۰۱

آزمونک اول

دستیاران آموزشی: آرین احدی نیا، امیررضا میرزایی

سوالات

نکات پیش از آزمون

۱. مدت زمان آزمون ۵۰ دقیقه با احتساب ۵ دقیقه زمان برای بارگذاری در سامانه کوئرا است. لطفا پس از ۴۵ دقیقه فرآیند بارگذاری را شروع فرمایید.
۲. مجموع نمرات پرسش‌ها از ۶۰ است اما در نهایت نمره شما از ۵۰ محاسبه خواهد شد. بنابرین دریافت ۵۰ نمره به منزله دریافت نمره کامل است.
۳. پاسخ سوالات را به صورت دست‌نویس بر روی کاغذ یا نوشت‌افزار دیجیتال (تبلت، قلم نوری و ...) یا به صورت تایپی بنویسید و به صورت یک فایل PDF در سامانه کوئرا آپلود کنید.
۴. تنها پاسخ نهایی شما در کوئرا تصحیح خواهد شد. در صورت آپلود مجدد پاسخ نهایی شما در سامانه کوئرا به صورت خودکار به آخرین پاسخ ارسالی تغییر خواهد کرد. لذا لطفا توجه کنید که پاسخ صحیح را به عنوان پاسخ نهایی در سامانه کوئرا قرار دهید.
۵. در تمام سوالات، میتوانید به سوالات دیگر یا بخش‌های دیگر همان سوال ارجاع دهید و از نتایج آن استفاده کنید؛ حتی اگر آن سوال یا بخش را حل نکرده باشید.
۶. در صورت مشاهده کپی، مطابق سیاست درس برخورد خواهد شد.

پرسش ۱ (۱۴ نمره) تنها به دو مورد از پرسش‌های زیر پاسخ کوتاه دهید.

(الف) (۷ نمره) اصطلاح Referential Transparency در زبان‌های تابعی به چه معناست؟

(ب) (۷ نمره) دلیل سریع‌تر بودن Direct Call Threading نسبت به Switch Threading چیست؟

(ج) (۷ نمره) چرا کامپایلرها از کد میانی استفاده می‌کنند؟

پرسش ۲ (۱۴ نمره) تابعی به نام merge بنویسید که در ورودی دو لیست مرتب از اعداد صحیح بگیرد و در خروجی آنها را در لیست مرتب ادغام کند.

ورودی نمونه ۱

```
(merge '(2 5 7 8 10) '(1 3 4 6 9))
```

خروجی نمونه ۱

```
'(1 2 3 4 5 6 7 8 9 10)
```

پرسش ۳ (۳۲ نمره) فرض کنید که n تابع داریم که به ترتیب باید روی ورودی مورد نظر اعمال شوند تا خروجی مورد نظر حاصل آید.

$$g(x) = f_1\left(f_2(f_3 \dots f_n(x))\right)$$

یا معادلا

$$g = f_1 \; o \; f_2 \; o \; f_3 \dots o \; f_n$$

بدیهی است که توابع f_1 تا f_n و تابع g همگی توابعی با یک ورودی و یک خروجی هستند. تضمین میشود که خروجی تابع f_{i+1} در دامنه ورودی تابع f_i قرار میگیرد. به بیان رسمی، بُرد تابع f_{i+1} زیرمجموعه دامنه تابع f_i خواهد بود.

(الف) (۱۴ نمره) تابعی به نام apply پیاده سازی کنید که در ورودی یک لیست از توابع f_1 الی f_n و یک مقدار x دریافت کند و در خروجی مقدار $g(x)$ را برگرداند. توجه کنید که امکان استفاده از توابع از پیش آماده Racket مانند fold که این کار را انجام می‌دهند در این سوال، وجود ندارد.

ورودی نمونه ۱

```
(apply (list
        (lambda (x) (* 2 x))
        (lambda (x) (+ 2 x))
        (lambda (x) (* 4 x)))
       1)
```

خروجی نمونه ۱

```
12
```

ورودی نمونه ۲

```
(apply (list - - -) 1)
```

خروجی نمونه ۲

```
-1
```

(ب) (۱۸ نمره) تابعی به نام super-function پیاده‌کنید که در ورودی یک لیست از توابع f_1 الی f_n را دریافت کند و در خروجی تابع g را برگرداند که در واقع همان $f_1 \; o \; f_2 \; o \; f_3 \; o \dots o \; f_n$ است. توجه کنید که خود خروجی تابع، تنها یک تابع مانند g است اما اگر این تابع بر روی یک مقدار مانند x فراخوانی شوند، مقدار $g(x)$ محاسبه خواهد شد.

ورودی نمونه ۱

```
(super-function (list
                 (lambda (x) (* 2 x))
                 (lambda (x) (+ 2 x))
                 (lambda (x) (* 4 x))))
```

خروجی نمونه ۱

```
#<procedure:...>
```

توضیح تکمیلی ۱

توجه کنید که خروجی تابع super-function خود یک تابع است بنابرین در خروجی تعریف یک تابع که به ظاهر بی‌معنا به نظر می‌رسد را مشاهده می‌کنیم.

ورودی نمونه ۲

```
((super-function (list
                  (lambda (x) (* 2 x))
                  (lambda (x) (+ 2 x))
                  (lambda (x) (* 4 x)))) 1)
```

خروجی نمونه ۲

```
12
```

توضیح تکمیلی ۲

همان‌طور که ملاحظه می‌فرمایید، اگر خروجی تابع super-function را یک تابع در نظر بگیریم و بر روی مقدار 1 آن را فراخوانی کنیم. حاصل g(1) بدست خواهد آمد.

ورودی نمونه ۳

```
(define g (super-function (list - - -)))

(g 2)
```

خروجی نمونه ۳

```
-2
```

توضیح تکمیلی ۳

اگر در محیط ابتدا تابع g را از خروجی super-function تعریف کنیم و سپس تابع g را بر روی مقدار 2 فراخوانی کنیم، g(2) حاصل خواهد آمد.