July 2^{nd} 2004 CMSC 10500-1 Midterm Duration: 2 Hours

Instructions

1. There is a total of 7 problems worth 80 points.
2. You will be marked out of 40 points.
3. You may solve as many problems as you wish.
4. The exam will be graded only based on whether your program works as it should or not.
5. Efficiency of your program will not be an issue as long as your program runs within a reasonable time (1 minute).
6. At the end of each problem some sample outcomes are given. You may use this to test the correctness of your program.
7. Please name your functions exactly as given in the problem. If you choose to define any additional functions you may name them as you wish.
8. Create a new directory called "midterm" in your home directory and store all your programs there.
9. Name your scheme files as "q<problem num>.scm".
10. To submit your programs type in the following in the unix shell

    `/home/gmkrishn/submit ~/midterm`.

    This will email the contents of the "midterm" directory to me. Just to be on the safe side, confirm with me that I have received your email.

Good Luck with the exam

Problems

1. Area of a triangle - 5 points

Write a scheme function `area`, which consumes three points on the plane (posn structures), and produces the area of the triangle defined by those points. You may use the following formula without proof:

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

where a, b, c represent the lengths of the three sides of the triangle and $s = \frac{a+b+c}{2}$ is the semi-perimeter of the triangle.

```
> (area (make-posn 0 0) (make-posn 3 0) (make-posn 3 4))
6
> (area (make-posn -4 0) (make-posn 4 0) (make-posn 0 8))
32
> (area (make-posn 5 24) (make-posn 5 19) (make-posn 7 26))
5
```

2. Preparing your taxes - 15 points

Consider the following scheme structure

```
;; Tax information of an individual
(define-struct tax-data (total-income tax-withheld num-exemptions))

;; Tax deduction numbers
(define std-deduction 5000)
(define exemption-amount 2500)

;; Tax return
;;    type is either 'DueToIRS or 'DueFromIRS
;;    amount is a non-negative number
(define-struct tax-return (type amount))
```

which represents the tax information of an individual.

1. The tax a person owes is a certain percentage of his **taxable income**. Figure 1 gives the appropriate percentage.
2. The **taxable income** is the total income, less **standard deductions** and **exemptions**.
3. The **standard deduction** is a fixed quantity given by the scheme variable `std-deduction`.

taxable income	tax percentage
≤ 7000	0
≤ 20000	10
≤ 50000	20
> 50000	30

Figure 1: Tax Slabs

4. The total **exemptions** is a fixed quantity (given by the scheme variable `exemption-amount`) multiplied by the number of exemptions claimed.

5. The quantity `tax-withheld` represents the amount of taxes already paid (with held from pay checks).

Write a scheme function `calc-tax-return` which consumes a `tax-data` structure and produces a `tax-return` structure. If the total tax is more than the amount already paid, then the person still needs to pay the difference to the IRS. If the reverse holds, then the person is due money from the IRS. For example,

```
> (calc-tax-return (make-tax-data 7000 2000 0))
(make-tax-return 'DueFromIRS 2000)
> (calc-tax-return (make-tax-data 55000 7000 4))
(make-tax-return 'DueToIRS 1000)
```

Comments: Preparing a tax return in real life is usually not so simple. Instead of standard deductions, one usually has itemized deductions which is a whole new can of worms!! Also there are complex rules which determine how many exemptions a person can claim.

3. More triangles - 10 points

Write a scheme function `configuration` which consumes three points on the plane and produces exactly one of the following symbols

1. `'CoLinear` if the three points lie on a straight line.

2. `'RightTriangle` if the three points form a right triangle.

3. `'Equilateral` if the three points form an equilateral triangle.

4. `'Isosceles` if the three points an isosceles triangle (but not an equilateral triangle).

5. `'NoneOfTheAbove` if none of the above hold.

You may assume that the three points are all different. Note that since numbers involved may be inexact, you may get unexpected results. Consider replacing checks like `(= a b)` with checks like `(<= (abs (- a b)) 0.0001)` which allow for a very small error.

```
> (configuration (make-posn -4 4) (make-posn 5 -5) (make-posn 0 0))
'CoLinear
> (configuration (make-posn 0 0) (make-posn 3 0) (make-posn 3 4))
'RightTriangle
> (configuration (make-posn -4 0) (make-posn 4 0) (make-posn 0 8))
'Isosceles
> (define a (* 4 (sqrt 3)))
> (configuration (make-posn -4 0) (make-posn 4 0) (make-posn 0 a))
'Equilateral
> (configuration (make-posn 0 0) (make-posn 1 1) (make-posn 3 4))
'NoneOfTheAbove
```

4. Approximating e - 10 points

Write a scheme function `approx-e` which consumes a number `n` and produces an approximate value for e using the formula:

$$\mathrm{e} \sim 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}$$

```
> (approx-e 7)
2.71825396...
> (approx-e 13)
2.7182818284467...
```

Hint: First write a function to calculate the factorial of a given number.

5. Number of distinct elements in a sorted list - 10 points

Write a scheme function `count-distinct` which consumes a list of increasing non-zero numbers and produces the number of different numbers present in the list. If the list is not sorted, you are free to produce whatever number you wish.

Hint: Since the list is sorted all occurrences of a given number occurs together. Hence it is enough to count the number adjacent positions which have different numbers.

```
> (count-distinct (cons 2 (cons 2 (cons 3 (cons 3 empty)))))
2
> (count-distinct (cons 1 empty))
1
> (count-distinct (cons 4 (cons 0 (cons 0 (cons -3 (cons -5 empty))))))
4
```

6. Shopping cart - 15 points

In this problem you implement a shopping cart. Your task is to write a scheme function `total-cost` which consumes a **shopping cart** and a **price list** and produces the total cost.

1. The total cost is the base cost together with a 8.75% tax.
2. The base cost is the sum of the costs of each item multiplied by the quantity.
3. Items which do not have a price mentioned are free!

The following scheme definition defines the structures you need.

```
;; item is a symbol
;; price is a number
(define-struct item-price (item price))

;; item is a symbol
;; qty (quantity) is a integer >= 0
(define-struct item-qty (item qty))
```

Thus a shopping cart is a list-of-item-qty and a price list is a list-of-item-price. Proceed as follows:

1. Write a function `item-cost` which consumes an instance of item-qty and a price list, and produces the cost of the given item (taking the quantity into account).
2. Write a function `base-cost` which consumes a list-of-item-qty (i.e. shopping cart) and a price list and produces the base cost for all the items.

Finally define `total-cost` which adds the tax.

```
> (define pricelist (cons (make-item-price 'LordOfTheRings 20.34)
                          (cons (make-item-price 'Pencil 0.75)
                          (cons (make-item-price 'HTDP 40.00)
                          (cons (make-item-price 'HarryPotterDVD 20.00)
                          (cons (make-item-price 'DaVinciCode 13.75)
                          (cons (make-item-price 'RuledNotebook 3.75)
                          (cons (make-item-price 'Pen10Pk 2.50)
                          (cons (make-item-price 'PhotoFrame 7)
                          empty)))))))))
> (define cart1 (cons (make-item-qty 'LordOfTheRings 1)
                      (cons (make-item-qty 'HTDP 1)
                      (cons (make-item-qty 'Pencil 10)
                      (cons (make-item-qty 'RuledNotebook 5)
                      (cons (make-item-qty 'DellComputer 10)
                      empty))))))
```

```
> (define cart2 (cons (make-item-qty 'HarryPotterDVD 1)
                      (cons (make-item-qty 'PhotoFrame 2)
                      (cons (make-item-qty 'TeddyBear 3)
                      (cons (make-item-qty 'RuledNotebook 5)
                      (cons (make-item-qty 'Pen10Pk 4)
                      empty))))))
> (total-cost cart1 prices)
94.166625
> (total-cost cart2 prices)
68.240625
```

7. Primality Testing - 15 points

In this problem you get to write a scheme function `factor` which consumes an integer ≥ 2 and returns a factor of that number ≥ 2, using the following outline:

1. Write a scheme function `factor?` which consumes two numbers `a` and `n` and checks if `a` is a factor of `n`, i.e. n/a has no remainder.

2. Write a scheme function `factor-in-range` which consumes three numbers `low`, `high` and `n` and produces some factor of `n` in the range `low` to `high` or `n` if `n` has no factor in the given range.

Finally observe that if `n` has a factor between `2` and `n`, then it has a factor between `2` and `(floor (sqrt n))` (you don't have to prove this). Use this observation to complete this problem.

```
> (factor? 6 42)
true
> (factor? 42 6)
false
> (factor-in-range 7 7 49)
7
> (factor-in-range 7 6 45)
45
> (factor-in-range 2 8 73)
73
> (factor 45)
3
> (factor 73)
73
```