Objectives

- Practice for Preliminary Examination 1
- Write and "run" code on paper

Value

This assignment is worth a maximum of 0 points. The assignment requires no submitted work.

Activities

Note: most of these involve writing or evaluating code, so you can do them in Scheme. However, you will not have a computer during the exam, so I suggest you try to determine values without the computer first; similarly try and write the functions as if you were doing it on paper before testing. Take 50 minutes to write your solutions on paper (assuming you are taking an exam) and 1 hour to test them using a computer.

1. Define a simple function to compute the following:
   (a) Define a function `USD-to-Bitcoin` which converts U.S. dollars to Bitcoins. Use this exchange rate: 1 Bitcoin = 389.943 USD.
   (b) Define a function `energy-from-mass` which uses Einstein's equation $E = mc^2$ to compute the energy equivalent of a mass m. Your function should take a single argument (m, the mass in kg) and return mc^2, where $c = 299,792,458\frac{m}{s^2}$. Use the `let` form to define the constant c. (Incidentally, with this choice of units, you will be computing the energy in Joules and your mass would be measured in kilograms.)
   (c) Define a function which takes the base and perpendicular height of a triangle and computes the area of that triangle (recall that the area is equal to $\frac{1}{2}b \times h$).
   (d) The SuperFresh supermarket chain needs a program to determine the value of all of the change in a cash register drawer. Define a function that, given the number of pennies, nickels, dimes and quarters, will calculate the total value of the change in the drawer. (Note that your function should take 4 arguments).
2. Define a recursive function to compute the following:
   (a) Observe that a repeating decimal can be calculated to an arbitrary number of decimal places using a recursive function. Write a function that takes one parameter, n, and calculates
   $$0.\bar{1} = 0.1111\ldots$$
   to n decimal places. Thus, your function, when called with 3, should return 0.111.
   Note: $\sum_{i=1}^{\infty} \frac{1}{10^i} = 0.\bar{1}$. This is the decimal expansion of the fraction 1/9.

(b) To amplify the previous problem, define a function that computes

$$1/22 = 0.0\overline{45} = 0.0454545\ldots$$

to n decimal digits of accuracy. How can you figure out if you should produce a 4 or a 5 in a given recursive step?
(Hint: There are a few ways to proceed. One is to use the fact that `(even? n)` returns true exactly when n is even. Another is to define a helper function with an argument that tells it(self) whether the next digit should be a 4 or a 5.) For practice, you might try to develop both.

(c) Consider the construction of a 3-dimensional pyramid using square blocks. A pyramid of height one is constructed with a single block; a pyramid of height two is constructed as shown in Figure 1, by building a foundation of 9 blocks and placing a pyramid of height one on the center. In general, we build a pyramid of height k in two steps:

1. Build a pyramid of height $k-1$.
2. Build a square foundation that's large enough so that when the pyramid of height $k-1$ is placed on top, the foundation protrudes by one box on each side.

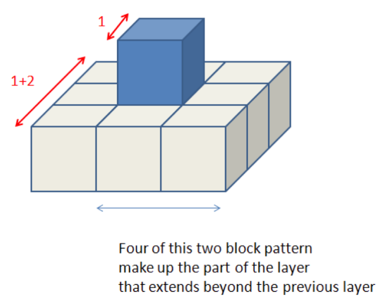

Figure 1: A pyramid of height 2.

The base of a pyramid of height 1 is 1x1; the base of a pyramid height 2 is 3x3. What should the base of a pyramid of height k be?

Once you have figured this out, define a Scheme function that computes the total number of blocks in a pyramid of height k.

3. An invasive vining plant spreads by branching and then rooting at its branch points, making a new plant. If there are fewer than 10,000 plants they spread in an "explosive growth phase" so that the number of plants *squares* at each time step. Once the population reaches 10,000, they spread by simply doubling during each time step.

   Consider a plant population that starts with 2 plants; letting p_t denote the number of plants at time step t, we then have that $p_0 = 2$. Our rules above state that for each $t \geq 0$,

$$p_t = \begin{cases} 2 & \text{if } t = 0, \\ p_{t-1}^2 & \text{if } p_{t-1} < 10000, \\ 2p_{t-1} & \text{if } p_{t-1} \geq 10000. \end{cases}$$

Write a recursive Scheme function to calculate the number of plants at a given time t. To really get it right, make sure that your function, when called to compute p_t, only computes p_{t-1} once (you may need a `let` statement for this... why?) Additionally, it seems that you will need the function `square`, that squares its input; define this function in a "private" way internal to the function that computes p_t.

4. Yawning is contagious. Suppose three tired students yawn (at the same time) in class, once every five minutes, starting when the lecture starts, no matter how riveting the lecture is. At every five minute time tick, the students previously yawning not only continue to yawn but also add to the number of yawners by infecting additional students with an infection rate of two new students per yawning student. These newly infected students yawn starting right away.

   So at time interval t_k the number of yawning students, y_k is $((1+2)y_{k-1})$.

   Write a recursive function that calculates how many yawns occur at any given five minute time tick. Show how you would use that function to calculate how many students are yawning when the lecture finishes 50 minutes later.

5. The integer 2 can be written as the sum of two squares: $2 = 1^2 + 1^2$; the integer 5 can likewise be written as the sum of two squares: $5 = 1^2 + 2^2$. You can check that 8 and 10 can also be represented this way.

   Write a function `sum-of-squares` so that `(sum-of-squares n)` returns #t if n can be written as a sum of squares and #f otherwise. You will probably need to define some "helper" functions.

6. Write a Scheme expression to evaluate 2^{256}, without using `expt`.
   (Hint: Of course, a possible solution is `(* 2 2 2 ... 2)` with 256 arguments. However, there are many more elegant solutions. One way to proceed is to suppose that you have found an expression `<E>` that evaluates to 2^{128}. Then the following expression evaluates to 2^{256}:

```
(let ((x128 <E>))
  (* x128 x128))
```

   Now, you can use this as the seed of a simple expression for 2^{256}.)

7. Higher-order functions:

   (a) In lecture you have seen a higher order function that will compute the summation of n terms of a function, f. Here is one for the product $f(0) \times f(1) \times f(n)$:

```
(define (product f n)
  (if (= n 0)
      (f 0)
      (* (f n) (product f (- n 1)))))
```

   i Use this product function to define a function that computes n!

   ii A function related to the factorial function is the product of all of the odd positive integers up to some odd integer n. This function is often referred to as odd double factorial. For an odd positive integer $n = 2k-1$, $k \geq 1$, odd double factorial is defined as $(2k-1)!! = \prod_{i=1}^{k}(2i-1)$. Define a Scheme function `odd-double-factorial` that uses `product`.
   Note: `odd-double-factorial` of an even number should be 0.

(b) Let f be a function that maps integers to integers. (The integers are the numbers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.) For such a function, define Δf to be the function

$$\Delta f(x) = f(x+1) - f(x)\,.$$

(Note that, given f, Δf is a *function*, not a number.)

Define a Scheme program which, given a function f, returns the function Δf. (The process of transforming f to Δf is sometimes called "taking the discrete derivative.")

This process has some interesting properties. Define the (degree one) polynomial $\ell(x) = 16x + 4$. Compute the discrete derivative using your function above and evaluate the resulting function at a few places (or plot it). What do you notice?

Now define the function $q(x) = 11x^2 - 4x + 11$, a quadratic function. Take the combinatorial derivative *twice* to yield the function

$$\Delta(\Delta(q))\,.$$

As above, evaluate this new function at a few places (or plot it). What do you notice?