Assignment 1
COMP 302 Programming Languages and Paradigms

Brigitte Pientka
MCGILL UNIVERSITY: School of Computer Science

Due Date: 19 September 2013

Your homework is due at the beginning of class on Sept 19, 2013. All code files must be submitted electronically, and your program must compile. The answer to Q0 should be submitted as a txt-file with the name `Q0.txt`; the answer to Q1 should be submitted as a txt-file with the name `Q1.txt`. For the remaining questions, please fill in the template code given on mycourses and submit the files without changing their names.

Q0 3 points Copyright and Collaboration policy. Read the copyright and collaboration policy of this course. For each of the three scenarios below, write a short paragraph explaining who is in violation with the copyright and collaboration policy of the course and why.

Scenario 1: Bob and Tom are working on question Q3 of the homework and have already spent 1h trying to get the code to compile and the deadline is drawing near. They decide to ask CS wiz Hanna who already completed the assignment for help. They send her their code and ask her to explain to them why it does not type-check. She replies: "You cannot use `+.` since `+.` is the addition operation on floating point numbers. To add two integers you should use +." Bob and Tom change their code and submit their fixed code.

Scenario 2: Bob is an excellent student and diligently writes down all his class notes in html form. He also types up all his homeworks and his solutions. This is an incredible resource, since they even contain the discussions and variations of examples which come up in class. Some of his fellow students request that he makes his notes publicly available, so that other students can profit from these notes. He feels he can help his fellow students to study better and decides to set up a public wiki for his CS course material including assignments and midterm with solutions.

Scenario 3: John and Matt agreed to be in one team and have started to work already on question 1 of the homework. Matt talks over skype with Abbey who is also in his class. He mentions that he and John are really stuck on Q4 of the COMP302 homework. Abbey wants to help and sends him her solution in OCaml. A week later, Bob and Matt submit the solution to Q4; Abbey also submits her homework.

Q1 7 points Compiling, Type-checking and evaluation/binding warm-up.

Consider the programs in the file `hw1-fixme.ml`. Try to compile the program either by typing into the caml-toplevel `#use ``hw1-fixme.ml'';;` (note `#use` is a command in OCaml) or in a shell `ocaml hw1-fixme.ml`.

Your task is: Explain step-by-step what errors when trying to compile the file `hw1-fixme.ml`; write down the error message you encounter by copy and pasting your errors in your file Q1.txt and state below each error how you fix it to proceed to the next error until your program is error free.

Q2 20 points Programming warm-up.

Q2.1 10 points The average can be computed follows: $\overline{x} = \frac{1}{n}\sum_{i=0}^{n} x_i = \frac{(x_1+...+x_n)}{n}$

Write a function `average: float list -> float` which accepts a list of floating point numbers and returns their average as a floating point number.

To illustrate, here are some test cases:

```
# average [1.0 ; 2.5 ; 3.0 ; 7.1 ];;
- : float = 3.4
# average [13.0 ; 25.2 ; 2.2 ; 27.4 ];;
- : float = 16.95
```

Note: Pay careful attention to conversions! You can convert an integer to a floating point number using the library function `float`.

Q2.2 10 points The standard deviation is computed as follows: $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$

In other words, the standard deviation can be calculated as follows:

1. For each value x_i calculate the difference between x_i and the average value $\overline{x}$.
2. Calculate the squares of these differences.
3. Find the average of the squared differences. This quantity is the variance σ^2.
4. Take the square root of the variance.

Write a function `stDev: int list -> real` which given a list of integers, computes the standard deviation as a real. For the empty list, `stDev` returns zero.

```
# stDev [1.0, 5.0, 2.0, 7.0];
- : float = 2.38484800354236404
# stDev [13.2 ; 25.0 ; 22.3 ; 27.8];;
- : float = 5.48059075282948
```

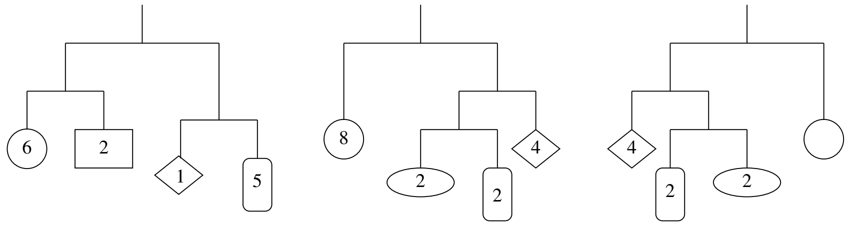
Note: Pay careful attention to conversions!

Q3 10 points Implement a function `psum:int list -> int list` that takes a list and computes the partial sums, i.e. the sums of all the prefixes.

Here are some cases illustrating the behavior.

```
# psum [1;1;1;1;1];;                # psum [];;
- : int list = [1; 2; 3; 4; 5]    - : int list = []
# psum [1;2;3;4];;                  # psum [9];
- : int list = [1; 3; 6; 10]      - : int list = [9]
```

Q4 30 points A mobile in art is a type of sculpture characterized by the ability to move when propelled by air currents or by touch. The parts of a mobile are usually carefully equilibrated parts. A mobile is usually made of wires, and objects (see picture below). The wires are assumed to be weightless, but objects have weights. The *weight* of a mobile is the sum of the weights of the objects attached to it. In the picture below, Mobile C has weight 16. A mobile is *balanced* if the mobiles attached at the ends of each wire are of the same weight. In the figure below, Mobile A is not balanced, while Mobile B, and C are balanced.



Mobile A Mobile B Mobile C

8

We can represent a mobile in OCaml using the following datatype.

type `mobile = Obj` **of** `int | Wire` **of** `mobile * mobile`

An object is represented by its weight (an integer) and a wire is represented by the two mobiles attached to its ends. The Mobile B for example is represented as

```
  Wire(Obj 8, Wire(Wire(Obj 2, Obj 2), Obj 4))
```

Q4.1 5 points Implement a function `weight:mobile -> int` which computes the total weight of a mobile.

Q4.2 10 points Implement a function `balanced:mobile -> bool`

Q4.3 10 points We can reflect a mobile about its vertical axis; for an object, the reflection is just itself; for a wire, we swap the positions of the two mobiles hanging off its end. For example, Mobile C is the reflection of Mobile B. Note reflection must be applied recursively. Implement a function `reflect:mobile -> mobile` which reflects a mobile.

We will now modify the representation of the mobile slightly and keep the weight information at the wire. The weight at the wire is the sum of the weights of each mobile attached to it.

```
type rmobile = RObj of int | RWire of rmobile * int * rmobile
```

Q4.4 5 points Implement a constant time function `rweight:rmobile -> int` which computes the total weight of a `rmobile`.

Q5 30 points In this exercise we explore the translation between binary numbers and natural numbers. First, we define a datatype for binary numbers.

```
type bnum = E | Zero of bnum | One of bnum
```

Binary numbers are represented in **reverse order**. For example,

Binary number	`bnum` representation	Natural number
110	`(Zero (One (One E)))`	6
0110	`(Zero (One (One (Zero e))))`	6
0101	`(One (Zero (One (Zero e))))`	5

Q5.1 15points Write a function `intToBin:int -> bnum` which translates an integer n ($n \geq 0$) into a binary number with no leading zeros.

Q5.2 15points Write a function `binToInt:bnum -> int` which translates a binary number into an integer.