Programmazione 1 — Tutorato 1

1. Scrivere una funzione `Sign(x)` che prende in input un numero x e ne restituisce il segno, ovvero restituisce:
$$\text{sign}(x) = \begin{cases} -1 & \text{se } x < 0, \\ 0 & \text{se } x = 0, \\ 1 & \text{se } x > 0. \end{cases}$$

2. Scrivere una funzione `Compose(f, g)` che prende in input due funzioni $f, g : \mathbb{R} \to \mathbb{R}$ e restituisce in output una funzione che calcola i due valori: $a = f(g(x))$ e $b = g(f(x))$. Per esempio:

```
h = Compose(lambda x: x**2, lambda x: 7/x)
print(h(2))
#Out[]: (12.25, 1.75)
```

3. Usando le `pairslist`, scrivere una funzione `Halve(As)` che prende in input una lista `As` e restituisce in output due liste, la prima contenente la prima metà di elementi di As, la seconda lista la seconda metà di elementi. Per esempio: `Halve((1,(2,(3,(4,(13,(21,None)))))))` deve restituire le due liste `(1,(2,(3,None)))`, `(4,(13,(21,None)))`.

4. Usando le `pairslist`, scrivere una funzione `Zip(As, Bs)` che prende in input due liste `As` e `Bs` e restituisce in output una lista di coppie, contenente le coppie degli i-esimi elementi delle due liste. Per esempio: `Zip((1,(2,(3,None))), (3,(4,(5,None))))` deve restituire la lista `((1,3), ((2,4), ((3,6), None)))`.

5. Usando le `pairslist`, scrivere una funzione `Pairs(As)` che prende in input la lista `As` e restituisce in output una lista di coppie, contenente le coppie degli $(i, i+1)$ per $i = 1, \ldots, n-1$, dove n è la lunghezza della lista. Per esempio: `Pairs((1,(2,(3,None))))` deve restituire la lista `((1,2), ((2,3), None))`.

6. Usando le `pairslist`, scrivere un predicato `IsSorted(As)` che restituisce `True` se la lista di numeri interi `As` è ordinata in modo decrescente, e `False` altrimenti.

7. Si consideri la funzione `FoldRight(P, As, v)` vista a lezione:

```
def FoldRight(P, As, v):
    if IsEmpty(As):
        return v
    return P(Head(As), FoldRight(P, Tail(As), v))
```

   Usando la funzione `FoldRight`, si scrivano le funzioni seguenti:

   (a) `And(Ls)`: prende in input una lista di valori booleani e ne calcola l'`and` complessivo

   (b) `Or(Ls)`: prende in input una lista di valori booleani e ne calcola l'`or` complessivo

   (c) `FoldFactorial(n)`: prende in input un numero intero n e calcola il fattoriale $n!$

8. Scrivere una funzione `Str2Int(s)` che prende in input una stringa `s` di sole cifre (numeri da 0 a 9) e converte la stringa in un numero intero in base 10. Ad esempio, la stringa "123" deve dare il numero $100 + 20 + 3 = 123$.

9. Scrivere una funzione `Int2Str(n)` che prende in input un numero intero n e lo converte in una stringa.

10. Scrivere un predicato `IsUpper(s)` che prende in input una stringa e restituisce `True` se la stringa contiene solo caratteri alfabetici (da 'A' fino a 'Z', alfabeto inglese) maiuscoli.

11. Scrivere un predicato `IsLower(s)` che prende in input una stringa e restituisce `True` se la stringa contiene solo caratteri alfabetici minuscoli (da 'a' fino a 'z', alfabeto inglese).