Corso di Informatica@Matematica
Prova scritta del 13 settembre 2017

Tempo a disposizione: ore 2.
Es. 3 e 5 necessari per la sufficienza.
In tutti gli esercizi è possibile definire funzioni ausiliarie.

1. (6 punti) La stringa `1111 1111` è la rappresentazione in complemento a due su 8 posizioni di un intero n. Qual è il valore di n? Qual è la rappresentazione di $n+1$ e di $n-1$?

2. (6 punti) Voglio invertire una lista (il primo elemento diventa l'ultimo, ecc.) lasciando però fermi gli elementi di indice dispari (si suppone 0 pari). È corretta la seguente funzione? Motivare.

```
def invpari(L):
    for i in range(0,len(L)//2,2):
        L[i],L[len(L)-1-i]=L[len(L)-1-i],L[i]
    return L
```

3. (7 punti) Si scriva una funzione `confronta(T1, T2)` che prende come parametri due stringhe `T1` e `T2` della stessa lunghezza e restituisce la lista degli `i` tali che `T1[i]` è uno dei caratteri `T2[i-1],T2[i],T2[i+1]` (se esistono tutti). Esempio: `confronta('la mamma in', 'la nonna ti'')` restituisce la lista: `[0, 1, 2, 7, 8, 9]`.

4. (6 punti) Cosa stampa il seguente frammento di programma Python? (si possono usare artifici grafici per descrivere l'output):

```
A = [0,1,2,3,4]
A[-1] = A
A[1] = [20,30,40]
B=A[:]
B[1][0]=100
print(A,B)
```

5. (7 punti) Si supponga di avere a disposizione le seguenti funzioni Python, che permettono di costruire e manipolare alberi binari (come sono realizzate non è significativo per l'esercizio):

```
def tree():              # Restituisce l'albero vuoto
def bint(a,l,r):          # Costruisce (restituisce) l'albero con radice etichettata a,
                         # figlio sinistro l e figlio destro r
def is_empty(a):         # Restituisce True sse a e' l'albero vuoto
def label(a):            # Restituisce l'etichetta della radice dell'albero a
def first_child(a):      # Restituisce il figlio sinistro dell'albero a
def second_child(a):     # Restituisce il figlio destro dell'albero a
```

Manipolando gli alberi solo con queste funzioni, si scriva una funzione `somma(a)` che preso un albero binario `a` etichettato con interi restituisce la somma delle etichette su quei nodi che tra i loro figli hanno *esattamente* una foglia (se `a` è l'albero vuoto, `somma(a)` restituisce 0). Esempio:

```
argom:     1                result: 17 (la somma di 2, 7 e 8)
          / \
         2   3
        / \   \
       4   5   6
      /         \
     7           8
      \           \
       9           1
```