Corso di Informatica@Matematica
Prova scritta del 11 Giugno 2019

Tempo a disposizione: ore 2.

In tutti gli esercizi è possibile definire funzioni ausiliarie. Fare gli esercizi **(1, 2) e (3, 4, 5)** su due fogli differenti. Il voto massimo è **24** la sufficienza è **13** . Il **voto finale** dell'esame si ottiene sommando i due voti sufficienti di scritto e laboratorio (studenti di matematica).

1. (5 punti) Uno studente del corso di Informatica si presenta all'esame dicendo di aver scritto una funzione TrovaPippo(P) che, preso come argomento una funzione Python P che non ha bisogno di argomenti (passata come stringa), restituisce True se l'esecuzione di P() stampa la stringa "Pippo"; restituisce False se l'esecuzione di P() non stampa la stringa "Pippo". Si dimostri che questo studente deve essere bocciato, perché una funzione come TrovaPippo non può esistere. Suggerimento: si consideri cosa succede se come P viene passata a TrovaPippo una funzione come la seguente, dove Q è un'altra, generica funzione senza argomenti che non stampa niente.

```
def P():
    Q()
    print("Pippo")
```

2. 5 punti) Scrivere in Python una funzione a valori booleani Suff(p,s) che presi come parametri due stringhe p e s, restituisce True sse p è un suffisso proprio di s. Il testo della funzione non può utilizzare l'operatore in; inoltre tutti i confronti devono avvenire tra singoli caratteri (non tra sottostringhe o slice).

3. (4 punti) Si applichi l'algoritmo di somma alla seguente coppia di numeri in complemento a due su 8 bit, discutendo eventuali overflow: `1000 1101`; `1111 0001`.

4. (5 punti) Cosa stampa il seguente programma Python?

```
class A:
    def __init__(self, xx, yy):
        self.x = xx
        self.y = yy

    def g(self):
        self.x = self.y-10
        return self.x

    def _h(self, v):
        return v*2

class B(A):
    def __init__(self, xx, zz):
        self.x = self._h(xx)
        self.y = self._h(zz)

    def f(self,delta):
        self.y = self.g() // delta


CC = B(20, 30)
CC.f(7)
print(CC.x, CC.y)
```

5. (5 punti) Si supponga di avere a disposizione la definizione della classe Python `tree` (vista a lezione) che introduce gli alberi binari etichettati mediante i seguenti costruttori e metodi (`t` è un albero):

```
tree()              # Restituisce l'albero vuoto
tree(a)             # Restituisce l'albero con una sola foglia/radice etichettata a
tree(a,l,r)         # Restituisce l'albero con radice etichettata a,
                    # figlio sinistro l e figlio destro r
t.label()           # Restituisce l'etichetta sulla radice dell'albero t
t.is_empty()        # Restituisce True sse t \'e l'albero vuoto
t.first_child():    # Restituisce il figlio sinistro dell'albero t
t.second_child():   # Restituisce il figlio destro dell'albero t
t.is_leaf():        # Restituisce True sse t \'e una foglia
```

Si scriva una funzione Python `leafsparent(t)` che restituisce la lista delle etichette presenti sui soli nodi di `t` i cui figli sono entrambi delle foglie (l'ordine nel quale le etichette sono raccolte non è importante ai fini dell'esercizio).

```
argom:     1          result: ['b',6] (o una sua qualsiasi permutazione)
          / \
         2   3
        / \   \
       4   a   6
      /       / \
     b       2   5
    / \
   5   1
```