1

(1/1 punto)

`BubbleSort()` è una funzione Python che implementa la versione più ottimizzata dell'algoritmo *bubble-sort*: prende in input una lista di interi e la modifica ordinando i suoi elementi in modo crescente. Sia `a` una lista di `n` interi, qual è il costo computazionale (caso peggiore) della seguente funzione?

```
def MultSort(a, n):
    for i in range(n):
        BubbleSort(a)
```

- Lineare
- Quadratico ✓
- Cubico
- Proporzionale a **nlog(n)**

2

(1/1 punto)

Qual è il costo computazionale della seguente funzione C?

```c
void f(int n){
    int i, j;
    for(i = 0; i < n; i++){
        j = i;
        while( j < n ){
            j++;
        }
        while( j > 0 ){
            j--;
        }
    }
}
```

- [x] n^2 ✓
- [ ] n
- [ ] n^3

3

(1/1 punto)

Esercizio 6

a e b sono due liste concatenate contenenti interi e d un dizionario, inizialmente vuoto, implementato con liste di trabocco. Gli elementi di d sono coppie (k, v) dove la chiave k è di tipo intero e la chiave v è di tipo puntatore. Vengono eseguite le seguenti operazioni:

- per ogni elemento x di a, la coppia (x, NULL) viene inserita in d;
- per ogni elemento x di b, la coppia (x, NULL) viene inserita in d.

Se a contiene n elementi e b ne contiene m, quanti elementi contiene d?

- meno di n+m+1 ✓
- n
- almeno n+m
- max(n,m)

4

(1/1 punto)

Sia a una stringa di lunghezza n > 1, cosa viene stampato dal seguente frammento di codice C?

```c
int n = strlen(a);
char *b = a+n/2;
*b = '\0';
printf("%ld\n", strlen(a)-strlen(b));
```

- un numero < 0
- un numero >0 ✓
- 0

5

(1/1 punto)

Sia **a** una lista di interi non negativi e **k** un intero > 16, cosa viene stampato al termine del seguente frammento di codice?

```
d = {}
for i in a:
    d[i] = i*'x'

print(d.get(k, 'x'*k))
```

- una stringa di lunghezza almeno **n**
- una stringa di lunghezza al più **k**
- una stringa di lunghezza **k** ✓
- una stringa o un intero

6

(1/1 punto)

Sia n un intero positivo e $0 \leq k < n$, qual è il valore di `len(a[k])`?

```
a = ['x']
for i in range(n):
    a.append('x'+a[-1])
```

- n
- k+1 ✓
- n+1
- k
- k-1
- n-1

7

(1/1 punto)

Sia a una lista concatenata contenente n interi ordinati in modo crescente, qual è il costo computazionale della ricerca di una chiave k in a?

- ordine di log(n)
- lineare ✓
- quadratico
- costante

8

(1/1 punto)

Sia **a** una stringa, quale tra le chiamate alla funzione enigma restituisce sempre **True**?

```
def enigma(x):
    if len(x) < 2:
        return True
    if x[0] != x[-1]:
        return False
    return enigma(x[1:-1])
```

- enigma(2*a)
- enigma(a[:2])
- enigma(a+a[::-1]) ✓

9

(1/1 punto)

Si consideri la seguente funzione C:

```
void f(int *x, int p){
    *(a+p) = 2*p;
}
```

Sia a un array di n>0 interi tale che a[i] = i per ogni indice i e sia 0≤k<n. Dopo l'invocazione di f(a,k) qual è il valore di a[k]?

- [x] 2k
- [ ] k
- [ ] indefinito

10

(1/1 punto)

Sia n un intero positivo ed f definita come segue, qual è il risultato di f(n)?

```
def f(n):
    a = list(range(n))
    a.append(list(range(n,2*n)))
    a += list(range(2*n, 3*n))
    return a[n]*a[n+1]
```

- una tupla
- un intero
- il prodotto di due liste
- una lista ✓