



Appello 15/06/23

PRETEST

- qual'è il valore della variabile n al termine del seguente frammento di codice c?

```
int n, a = 9;
int *b = &a;
a++;
n = *b +1;;
```

Risposta data: indefinito —> perché il professore ha fatto un errore di battitura sull'ultima riga mettendo due ;;

risposta corretta: 11 —> perché b punta all'indirizzo di a, e al momento che a viene incrementata b assume lo stesso valore

- sia L una lista concatenata implementata in c: ogni nodo contiene un intero ed il puntatore al nodo successivo oppure NULL se è l'ultimo nodo; L è una struct che contiene il puntatore al primo nodo della lista ed il numero di n di elementi della lista. Gli interi della lista sono ordinati in modo crescente. Qual è il costo (computazionale) di un algoritmo efficiente che risolva il problema della ricerca di una chiave nella lista?

risposta data: O(n) nel caso migliore

risposta corretta: O(n) nel caso peggiore

—> la differenza è molto sottile, essenzialmente una lista concatenata permette l'accesso ad un elemento in posizione x solo se si visitano gli x-1 elementi precedenti; da questo si evince che il caso peggiore (ovvero la chiave da cercare si trova nell'ultima posizione) il costo computazionale è di O(n) nel caso peggiore.

- qual'è il valore di b al termine del seguente codice?

```
a = [4,3,2,1]
for x in a:
    try:
        b += x
    except NameError:
        b = 1
```

risposta data (corretta): 7 —> questo perché nella prima iterazione (ove $x = 4$) il codice cattura un errore del tipo NameError (b non è stato dichiarato) quindi nella prima iterazione $b = 1$, da lì in poi vengono sommati 3,2,1.

4. si consideri la seguente struttura:

```
d = {3: 'python'}
```

quale tra le operazioni elencate non è ammissibile per d?

risposta data: $d[(1,2)] = 3$

risposta corretta: $d[[1,2]] = 3$

—> le liste non vengono accettate come chiavi perché sono degli elementi mutabili (possono essere modificate) il che causerebbe inconsistenza di dati se venissero accettate.

5. cosa viene stampato dal seguente frammento di codice c?

```
char a[] = "python";
char b = malloc(100);
int i = 0;
while (a[i] != '\0') {
    b[i] = a[i];
    i++;
}

printf("...%d\n", strlen(b));
```

risposta data (corretta) un intero indefinito —> questo perché b ha 100 byte di memoria allocati, e non viene impostato il carattere di fine stringa ('\0'); questo implica che strlen(b) non può essere stimato con certezza.

6. sia L una lista concatenata implementata in c: ogni nodo contiene un intero ed il puntatore al nodo successivo oppure NULL se è l'ultimo nodo; L è una struct che contiene il puntatore al primo nodo della lista ed il numero di n di elementi della lista. sia a un array di m interi. quale sarebbe la complessità temporale di una funzione che prenda in input L ed a ed inserisca gli elementi di a in fondo a L?

risposta data (corretta): $O(\max(m,n))$ —> una volta trovato l'ultimo nodo della Lista L dobbiamo solamente aggiungere gli m elementi in coda.

codice extra:

```
nodo *nd = search(L, n-1); //restituisce l'ultimo elemento
int i = 0;
while( i < m) {
    nd->v = a[i]; //assegna il valore i-simo dell'array a a
    nd = nd->next; //passo al prossimo nodo
    i++;
}
```

7. cosa viene stampato dal seguente frammento di codice?

```

a = [5,4,3,2,1]
b = [(a[i], a[-1 -i] for i in range(len(a))]
c = [ x+y for x,y in b]
print(sum(c))

```

risposta data (corretta): 30 —> perchè in b vengono create tutte tuple di 2 elementi, e per ogni tupla di 2 elementi la rispettiva somma è 6, quindi nella lista c vengono salvati 5 interi con valore 6, che sommati tra loro da 30. n.b: quando si accede ad una lista con dei valori negativi bisogna partire sempre dall'ultimo elemento fino al numero specificato.

8. assumendo che un puntatore occupi 8 byte ed un int 4 byte, quale delle seguenti affermazioni è vera dopo l'esecuzione del seguente frammento di codice?

```

int *a = malloc(2*sizeof(a));
int b[8] = {0};
int size_a = sizeof(a);
int size_b = sizeof(b);

```

risposta data (corretta): $\text{size_b} == 4 * \text{size_a}$ —> la sizeof di un puntatore rimane sempre 8 byte (indirizzo puntato) mentre la sizeof di un array è il valore del tipo di dato * il numero di elementi; da semplici calcoli si capisce la risposta

9. a e b sono due liste di dimensione rispettivamente n_a ed n_b . Qual è il costo nel caso medio di un algoritmo efficiente che computi la lista delle chiavi che sono sia di a che di b?

risposta data: $O(n_a^{**2} + n_b^{**2})$

risposta corretta: $O(\max(n_a, n_b))$

—> basta leggere la domanda: "lista delle chiavi che sono SIA di a che di b". (lho buttata a caso la mia risposta stavo nel pallone)

10. Sia a una lista python di n interi ordinati dal più piccolo al più grande. qual è il costo di una funzione che inserisca un nuovo elemento in a mantenendone la proprietà dell'ordinamento?

risposta data: $O(\log n)$

risposta corretta: $O(n)$

—> qui sono stato troppo ottimista, nel caso peggiore una funzione del genere scandisce la lista fino all'ultimo elemento per trovare la posizione del nuovo elemento; si può quindi concludere che la complessità temporale sia $O(n)$

ESERCIZI

- 1) C

Si scriva una funzione che elimini da una stringa in input tutti i caratteri non alfabetici. La funzione abbia il seguente prototipo.

```
#include <stdio.h>

void remove_non_alpha(char *a);

void main()
{
    char a[] = "pr$€!o=gramm?^azione";
    remove_non_alpha(a);
    printf("La stringa senza caratteri non alfabetici e': %s\n", a);
}

void remove_non_alpha(char *a) {
    int i = 0, j = 0;
    while (a[i] != '\0') {
        if (((a[i] >= 'a' && a[i] <= 'z') || (a[i] >= 'A' && a[i] <= 'Z'))) {
            a[j] = a[i];
            j++;
        }
        i++;
    }
    a[j] = '\0';
}
```

2) Python

Si scriva una funzione, denominata intersection, che prenda in input due dizionari d0 e d1 e restituisca una lista contenente le chiavi che sono in de ed in di.

NB: Sono ammesse soltanto le strutture dati trattate e lezione ovvero list, tuple, dict e str.

```
def intersection(d0, d1):
    out = []
    for x in d0:
        if x in d1:
            out.append(x)

    return out

d0 = {'a': 1, 'b': 2, 'c': 3}
d1 = {'a': 1, 'b': 2, 'd': 4}
print(d0)
print(d1)
print(intersection(d1, d0))
```

Il costo computazionale di entrambi gli esercizi è O(n)