Appello di Programmazione e Algoritmi 1
16 settembre 2021

Esercizio 1

Cosa stampa il seguente programma Java?

```
public class Main {
public static void main(String[] args) {
                    System.out.println(calcola(2));
                    System.out.println(calcola(10));
                    System.out.println(calcola(5));
                    }
                static int prova (int x){
                    if (x<0) return x;
                    if (x%2==0) return calcola(x-1) + x;
                    return calcola(x-3) - x;
                  }
}
```

NON SARANNO CONSIDERATE VALIDE LE RISPOSTE NON GIUSTIFICATE DAI CALCOLI EFFETTUATI PER OTTENERE IL RISULTATO

Cosa stampa il seguente frammento di codice Java?

```
boolean termina=false;
int k=14,i;
for (i=0; i<9 && ! termina; i++) {
                    if (i*i>k){ termina=true;}
                    else {System.out.println ("punteggio = " + (k*i));
                    k=k-i;}
                }
System.out.println ("valore di k = " + k);
```

NON SARANNO CONSIDERATE VALIDE LE RISPOSTE NON GIUSTIFICATE DAI CALCOLI EFFETTUATI PER OTTENERE IL RISULTATO

Esercizio 2

Scrivere un metodo in Java

static int[] estraiRipetuti (int[] a)

che preso in input un array **a** di elementi interi, restituisce un nuovo array in cui sono presenti tutti e soli gli elementi di **a** che compaiono più di una volta in **a** stesso, nello stesso ordine in cui compaiono in **a**. Ad esempio: se a={5, 7, 9, 5, 8, 5,10, 3, 8} il metodo dovrà restituire {5, 5, 8, 5, 8} (ovvero tutte le occorrenze di 5 e 8, che sono gli elementi che compaiono più di una volta, nell'ordine in cui compaiono in a). Altro esempio, se **a**={1, 2, 3, 1, 5, 2} il metodo restituirà {1, 2, 1, 2}.

Si analizzi la complessità temporale del metodo proposto (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore*).

Esercizio 3

Scrivere un metodo in Java

static int[] finoA (int[] a, int k)

che preso in input un array **a** di elementi interi, restituisce un nuovo array in cui sono presenti tutti e soli gli elementi di **a**, a partire da quello iniziale dell'array **a** stesso fino a quello presente nell'indice k compreso. Ad esempio: se **a**={5, 7, 9, 5, 8, 5,10, 3, 8} e **k**=3, il metodo dovrà restituire {5, 7, 9, 5}, (ovvero tutti gli elementi che occorrono in **a**, a partire da quello di indice 0 fiano all'elemento di indice 3 incluso). Altro esempio, se **a**={1, 2, 3, 1, 5, 2} e **k**=0 il metodo dovrà restituire {1} (gli elementi di **a**, a partire da quello di indice 0 fino all'elemento di indice 0 compreso). Nel caso in cui k non sia un indice ammissibile per **a**, il metodo dovrà restituire null.
Si analizzi la complessità temporale del metodo proposto

Esercizio 4

Si consideri il seguente array di numeri interi:
`{20, 17, 22, 10,6, 23, 8, 11, 3, 21, 14}`
Mostrare **passo-passo l'esecuzione del merge-sort** per ordinare l'array in modo non decrescente.
Si descriva la complessità computazionale del merge-sort (anche scrivendo e risolvendo la ricorrenza T(n)).

[Esercizio di Laboratorio] SOLO PER GLI STUDENTI CLEII che hanno svolto il progetto dell'AA 2020/2021 In relazione al progetto di laboratorio, si implementino i seguenti due metodi:

- boolean almenoMeta(boolean[] album)

che restituisce true se nell'album passato come parametro sono presenti almeno metà delle figurine, false altrimenti;

- int[] figurineDaScambiare(boolean[] album1, boolean[] album2)

restituisce un array contenente gli indici di tutte le figurine presenti in album1 e che mancano in album2. Ad esempio, dati in input:

album1 = {false, false, true, true, false, true, false, true};
album2 = {false, true, true, false, false, true, false, false};

il metodo restituisce l'array {3,7}. Si assuma che album1 e album2 abbiano la stessa lunghezza.