Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi I A.A. 2018/2019

Compito n° 1

Appello del 10 gennaio 2019

Esercizio 1 (8 punti)

Si consideri il seguente programma Java:

```
public class Main {
	public static void main(String[] args) {
		int[] a = {10, 15, 7, 20};
		System.out.println(enigmaIt(a));
		System.out.println("---");
		System.out.println(enigmaRic(6, 2));
		System.out.println(enigmaRic(10, 3));
		System.out.println(enigmaRic(1003, 4));
	}

	static int enigmaIt(int[] a) {
		int ris = 0;
		for (int i = 0; i < a.length - 1; i++) {
			if (a[i] < a[i + 1]) {
				ris += a[i + 1];
			} else {
				ris -= a[i];
			}
		}
		return ris;
	}

	static int enigmaRic(int x, int y) {
		if (x < y)
			return x;
		return enigmaRic(x - y, y);
	}
}
```

1.1 (4 punti) Cosa stampa il programma?

1.2 (4 punti) Si analizzi la complessità computazionale del metodo `enigmaIt`.

Esercizio 2 (10 punti)

(7 punti) Scrivere un metodo (in Java o in pseudocodice)
`static boolean sommaDiDue (int[] a, int x)`
che, preso come parametro un array **a** di numeri interi ed un intero **x**, restituisce *true* se e solo se esistono due posizioni i e j di a tali che x=a[i]+a[j]. Se **a** vale **null**, viene restituito false. Il metodo **non** deve modificare l'array a.
(3 punti) Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 7*), dove n è la lunghezza dell'array a in input.
Ad esempio, se **a={5, 7, 3, 2, 9, 8, 10, 8}** e **x=9** il metodo deve restituire *true*, mentre se **x=-3** il metodo deve restituire *false*.

Esercizio 3 (8 punti)

a. Mostrare **l'heap di minimo** che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato le chiavi **5, 32, 8, 45, 12, 80, 9, 7, 43** (mostrare l'heap che si ottiene dopo l'inserimento di ogni chiave);
b. Rimuovere per 3 volte la chiave minima dall'heap, mostrando l'heap che si ottiene dopo ogni estrazione;
c. Inserire nell'heap così ottenuto le chiavi **3, 72, 6.**

Esercizio 4 (10 punti)

Si considerino i tipi di dato

```
class Esame {
      int matricola;
      int voto;
}
```

```
class Appello {
      Esame[] esami;
}
```

che rappresentano rispettivamente un esame di uno studente (con i campi matricola e voto) e un appello composto da diversi esami, contenuti in un array.
Attenzione: bisogna gestire tutti i casi in cui i tipi riferimento coinvolti possono valere null.

- **(5 punti)** Scrivere un metodo
  `static int contoVotoAlmenoUguale (Appello a, int voto)`
  che, presi come parametri un Appello **a**, restituisce il numero di studenti che hanno conseguito un voto almeno uguale a **voto**.
- **(5 punti)** Scrivere un metodo
  `static int[] estrai (Appello a, int voto)`
  che, facendo uso del metodo *contoVotoAlmenoUguale*, preso come parametro un Appello **a**, crea e restituisce un array contenente tutte e sole le matricole che hanno riportato un voto almeno pari a **voto** all'appello.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Le risposte al primo esercizio devono essere date direttamente nel riquadro di questo foglio.
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.

Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi I A.A. 2018/2019

Compito n° 2

Appello del 10 gennaio 2019

Esercizio 1 (8 punti)

Si consideri il seguente programma Java:

```
public class Main {
	public static void main(String[] args) {
		int[] a = {12, 16, 6, 15};
		System.out.println(enigmaIt(a));
		System.out.println("---");
		System.out.println(enigmaRic(5, 2));
		System.out.println(enigmaRic(11, 4));
		System.out.println(enigmaRic(1003, 5));
	}

	static int enigmaIt(int[] a) {
		int ris = 0;
		for (int i = 0; i < a.length - 1; i++) {
			if (a[i] < a[i + 1]) {
				ris += a[i + 1];
			} else {
				ris -= a[i];
			}
		}
		return ris;
	}

	static int enigmaRic(int x, int y) {
		if (x < y)
			return x;
		return enigmaRic(x - y, y);
	}
}
```

1.1 (4 punti) Cosa stampa il programma?

1.2 (4 punti) Si analizzi la complessità computazionale del metodo `enigmaIt`.

Esercizio 2 (10 punti)

(7 punti) Scrivere un metodo (in Java o in pseudocodice)
`static boolean sommaDiDue (int[] a, int x)`
che, preso come parametro un array **a** di numeri interi ed un intero **x**, restituisce *true* se e solo se esistono due posizioni i e j di a tali che x=a[i]+a[j]. Se **a** vale **null**, viene restituito false. Il metodo **non** deve modificare l'array a.
(3 punti) Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo 4 punti su 7*), dove n è la lunghezza dell'array a in input.
Ad esempio, se **a={5, 7, 3, 2, 9, 8, 10, 8}** e **x=9** il metodo deve restituire *true*, mentre se **x=-3** il metodo deve restituire *false*.

Esercizio 3 (8 punti)

d. Mostrare **l'heap di minimo** che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato le chiavi **5, 32, 80, 9, 7, 43, 8, 45, 12** (mostrare l'heap che si ottiene dopo l'inserimento di ogni chiave);
e. Rimuovere per 3 volte la chiave minima dall'heap, mostrando l'heap che si ottiene dopo ogni estrazione;
f. Inserire nell'heap così ottenuto le chiavi **3, 72, 6.**

Esercizio 4 (10 punti)

Si considerino i tipi di dato

```
class Esame {
      int matricola;
      int voto;
}
```

```
class Appello {
      Esame[] esami;
}
```

che rappresentano rispettivamente un esame di uno studente (con i campi matricola e voto) e un appello composto da diversi esami, contenuti in un array.
Attenzione: bisogna gestire tutti i casi in cui i tipi riferimento coinvolti possono valere null.

- **(5 punti)** Scrivere un metodo
  `static int contoVotoAlPiuUguale (Appello a, int voto)`
  che, presi come parametri un Appello **a**, restituisce il numero di studenti che hanno conseguito un voto al più uguale a **voto**.
- **(5 punti)** Scrivere un metodo
  `static int[] estrai (Appello a, int voto)`
  che, facendo uso del metodo *contoVotoAlPiuUguale*, preso come parametro un Appello **a**, crea e restituisce un array contenente tutte e sole le matricole che hanno riportato un voto al più pari a **voto** all'appello.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **100** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- Le risposte al primo esercizio devono essere date direttamente nel riquadro di questo foglio.
- Durante la prova scritta **non** è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- È possibile consultare appunti, libri e dispense.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**
- Mettere in vista sul banco un valido documento di identità.