Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2019/2020

Appello del 18 febbraio 2020

Esercizio 1 [8 punti]

Si consideri il seguente programma Java:

```
class A {
	protected int n;

	public A(int n) {
		this.n=n;
	}

	public int metodo (int a) {
		return a+n;
	}
	public double metodo (double b) {
		return b+n+0.5;
	}
}

class B extends A {
	public B(int n) {
		super (2*n);
	}
	public int metodo (int a) {
		return a+n+5;
	}
	public double metodo (double b) {
		return b+n+2.3;
	}
}

public class MainClass {
	public static void main(String[] args) {
		A a = new A(1);
		B b = new B(2);
		A c = b;
		System.out.println (c.metodo(4));
		System.out.println (b.metodo(a.metodo(2.0)));
		System.out.println (b.metodo(a.metodo(2)));
		System.out.println (c.metodo(c.metodo(0.4)));
	}
}
```

Dire cosa stampa il programma, giustificando la risposta mostrando in particolare:

- le firme associate a tempo di compilazione e di esecuzione ad ogni chiamata di metodo
- l'evoluzione della memoria nella parte heap.

Esercizio 2 [12 punti]

Si scrivano le classi Java per gestire *il bilancio in un condominio*.
Si scriva una classe astratta **Spesa** con

- una variabile di istanza **descrizione** (tipo String, private);
- una variabile di istanza **importo** (tipo double, private);

Scrivere i seguenti metodi di istanza:

- un costruttore che crea una spesa dati **descrizione** e **importo**;
- metodi pubblici accessori **`getDescrizione()`**, **`getImporto()`**;
- metodo astratto **`getRiparto(Condomino c)`** che calcola l'ammontare che il condomino c deve pagare per la spesa.

La classe Spesa è estesa dalle sottoclassi **SpesaGenerale** e **SpesaScala,** per le quali vanno scritti opportunamente i costruttori che richiamano il costruttore della classe Spesa e va implementato il metodo **`getRiparto(Condomino c)`** tenendo conto del fatto che un'istanza di SpesaGenerale va ripartita secondo i millesimiProprietà, mentre un'istanza di SpesaScala secondo i millesimiScala (la formula per ripartire una spesa **s** per un condomino che ha **x** millesimi è $s\frac{x}{1000}$).

Si assuma, senza scrivere nulla, l'esistenza di una classe **Condomino** con

- una variabile di istanza **nome** (tipo String, final, private)
- una variabile di istanza **cognome** (tipo String, final, private)
- una variabile di istanza **millesimiProprietà** (tipo int, final, private)
- una variabile di istanza **millesimiScala** (tipo int, final, private)
- un costruttore che prende tutti e 4 i parametri sopra descritti;
- metodi pubblici accessori per tutti e 4 i parametri sopra descritti.

Si assuma, senza scrivere nulla, l'esistenza di una classe **Versamento** con

- una variabile di istanza **eseguitoDa** (tipo Condomino, final, private)
- una variabile di istanza **importo** (tipo double, private);
- un costruttore che prende tutti e 2 i parametri sopra descritti;
- metodi pubblici accessori per tutti e 2 i parametri sopra descritti.

Si progetti infine una classe **Condominio** con

- una variabile di istanza **condomini** (tipo ArrayList<Condomino>, final, private);
- una variabile di istanza **spese** (tipo ArrayList<Spesa>, final, private);
- una variabile di istanza **entrate** (tipo ArrayList<Versamento>, final, private);

Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea ArrayList vuoti per condomini, spese e entrate.
- un metodo **public double getDovuto (Condomino c)** che restituisce il totale degli importi dovuti dal condomino c per tutte le spese presenti nell'ArrayList spese.
- un metodo **public double getPagato (Condomino c)** che restituisce il totale degli importi pagati dal condomino c e registrati nell'ArrayList entrate.
- un metodo **public ArrayList<Condomino> getMorosi ()** che restituisce un ArrayList contenente tutti e soli i condomini che non hanno pagato a sufficienza per coprire i pagamenti dovuti.

Regole per lo svolgimento della prova scritta:

- Per svolgere il compito si hanno a disposizione **120** minuti.
- Scrivere **subito** nome, cognome, matricola su OGNI FOGLIO (**compreso questo**).
- 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.

Esercizio 3 (6 punti)

Dato un intero positivo x ed un array a di n interi positivi, la seguente ricorrenza ritorna il valore **true** se e solo se esistono o meno indici $i_1, i_2, \ldots, i_k$ (tra loro distinti) nell'array a tali che:

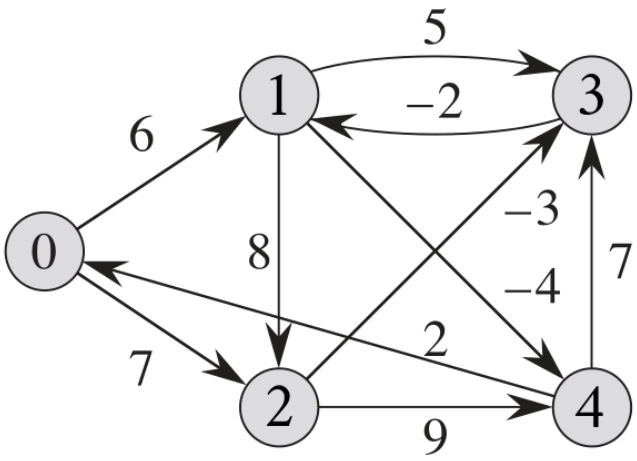
$$\sum_{j=1}^{k} a_{i_j} = x$$

In particolare, F(i,x) è la soluzione al problema che richiede di formare l'intero x utilizzando i primi i elementi dell'array a.

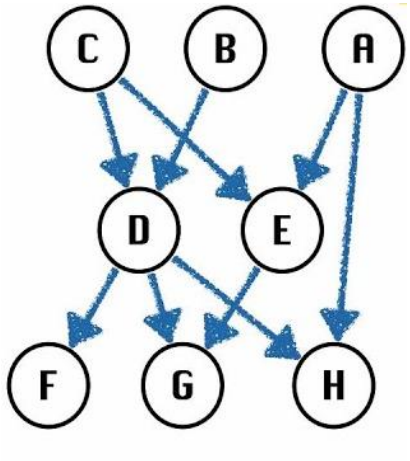
$$F(i,x) = \begin{cases} true & se\ x = 0 \\ false & se\ x > 0\ e\ i = 0 \\ F(i-1, x - a[i-1])\ or\ \ F(i-1,x) & altrimenti \end{cases}$$

a) **[3 punti]** Si fornisca una implementazione ricorsiva in Java dell'algoritmo proposto.
b) **[3 punti]** Si fornisca una implementazione ricorsiva con **memoization** in Java dello stesso algoritmo.

Esercizio 4 [14 punti]



a. **[7 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Bellman-Ford** per calcolare i cammini minimi a partire dalla sorgente **0.** Dire, giustificando la risposta, se il grafo possiede cicli di peso negativo.



b. **[7 punti]** Calcolare un ordinamento topologico usando il metodo della visita in profondità, evidenziando per ogni nodo i timestamp di inizio e fine visita.