Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2019/2020

Appello del 20 gennaio 2020

Esercizio 1 (8 punti)

Si consideri il seguente programma Java:

```
class A {
	int n;

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

	public int getN() {
		return n;
	}

	public A metodo (A a) {
		A ris = new B (n+ a.n);
		return ris;
	}

	public B metodo (B b) {
		B ris = new B (n + b.n + 1);
		return ris;
	}
}
class B extends A {
	public B(int n) {
		super (2*n);
	}

	public A metodo (A a) {
		A ris = new A (n + a.n + 2);
		return ris;
	}
}
public class MainClass {
	public static void main(String[] args) {
		A a = new A(3);
		B b = new B(2);
		A c = new B(1);
		System.out.println (c.metodo(a).getN());
		System.out.println (c.metodo(a.metodo(b)).getN());
	}
}
```

Dire cosa stampa il programma, giustificando la risposta mostrando le firme associate a tempo di compilazione ad ogni chiamata di metodo e l'evoluzione della memoria (stack e heap).

Esercizio 2 (12 punti)

Si scrivano le classi Java per gestire il sistema delle contravvenzioni per infrazioni stradali in un comune.
Si scriva una classe astratta **Personale** con

- una variabile di istanza **nome** (tipo String, private);
- una variabile di istanza **cognome** (tipo String, private);
- una variabile di istanza **matricola** (tipo int, final, private).

Implementare i seguenti metodi di istanza:

- un costruttore che crea un Vigile dati **nome, cognome e matricola**;
- metodi pubblici accessori **`getNome()`**, **`getCognome(), getMatricola()`**;

La classe Personale è estesa dalle sottoclassi **Vigile** e **Ausiliario,** per le quali vanno scritti opportunamente i costruttori che richiamano il costruttore della classe Personale e il metodo pubblico **`boolean equals (Object o)`** che restituisce true se e solo se o è un oggetto della stessa classe avente matricola uguale a quella di this (si noti che possono esistere un vigile e un ausiliario con la stessa matricola, che chiaramente **non** sono da considerare uguali).

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

- una variabile di istanza **descrizione** (tipo String, final, private)
- una variabile di istanza **puntiPatente** (tipo int, final, private)
- una variabile di istanza **importo** (tipo double, final, private)
- una variabile di istanza **targa** (tipo String, final, private)
- una variabile di istanza **autore** (tipo Personale, final, private);
- un costruttore che prende tutti e 5 i parametri sopra descritti;
- metodi pubblici accessori per tutti e 5 i parametri sopra descritti.

Si progetti infine una classe **ComandoMunicipale** con

- una variabile di istanza **persone** (tipo ArrayList<Personale>, final, private);
- una variabile di istanza **multe** (tipo ArrayList<Multa>, final, private);

Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea arraylist vuoti per persone e multe.
- un metodo **public double getImporto (int i)** che restituisce il totale degli importi delle multe fatte dal vigile o ausiliario in posizione i dell'arraylist persone (si sfrutti il metodo equals scritto per le classi Vigile e Ausiliario);
- un metodo **public Personale getMigliore ()** che restituisce il Vigile o Ausiliario che ha fatto multe per il più alto importo;
- un metodo **public boolean vigiliVsAusiliari ()** che restituisce true se e solo se il totale degli importi delle multe fatte dai vigili è superiore a quello relativo a multe fatte dagli ausiliari.

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 (10 punti)

Dato un intero positivo x ed un array a di n interi positivi, progettare un algoritmo di programmazione dinamica che ritorni 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:

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

Esempio: Si consideri a= {4, 7, 9, 2). È possibile ottenere 13 con tre indici 0,1 e 3 (ovvero a[0]+a[1]+a[3]=13).

Suggerimento: indicare con F(i,y) la soluzione al problema che richiede di formare l'intero y utilizzando i primi i elementi dell'array a.

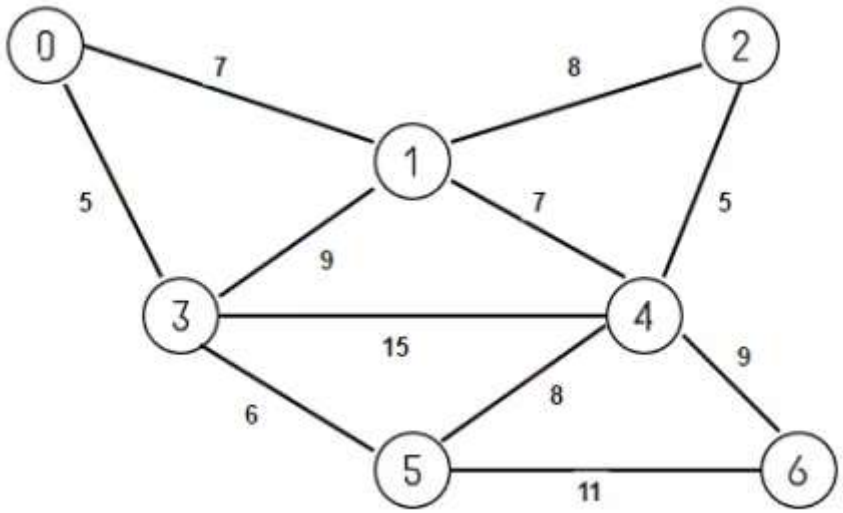
$$F(i,y) = \begin{cases} true & se\ y = 0 \\ false & se\ y > 0\ e\ i = 0 \\ ??\ or\ ?? & altrimenti \end{cases}$$

Nella terza riga (caso ricorsivo) bisogna considerare due casi: il caso in cui l'i-esimo elemento di a (cioè a[i-1]) concorre a formare y, e il caso in cui non concorre.

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

Esercizio 4 (10 punti)

Si consideri il grafo non diretto in figura.



a. **[5 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Kruskal** per il minimo albero ricoprente.
b. **[5 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Prim** per il minimo albero ricoprente a partire dal nodo sorgente **6**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.