Cognome e Nome _____

Matricola _____

Programmazione e Algoritmi 2 A.A. 2019/2020

Appello del 4 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 getN() {
		return n;
	}
	public A metodo (A a) {
		A ris = new B (n+ a.n);
		n = ris.n+1;
		return ris;
	}
	public B metodo (B b) {
		B ris = new B (n + b.n + 1);
		n = ris.n-1;
		return ris;
	}
}
class B extends A {
	public B(int n) {
		super (2 + n);
	}
	public A metodo (A a) {
		A ris = new B (n + a.n + 2);
		n = ris.n;
		return ris;
	}
}
public class MainClass {
	public static void main(String[] args) {
		A a = new A(1);
		B b = new B(2);
		A c = new B(3);
		System.out.println (c.metodo(b).getN());
		System.out.println (b.metodo(a.metodo(b)).getN());
		System.out.println (c.metodo(c.metodo(a)).getN());
	}
}
```

Dire cosa stampa il programma, giustificando la risposta e 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 vogliono gestire, in Java, gli ordini di un negozio di fast food.

Si assuma, senza scrivere nulla, l'esistenza di una classe **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).

e i seguenti metodi di istanza:

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

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

- una variabile di istanza **descrizione** (tipo String, private)
- una variabile di istanza **costoUnitario** (tipo double, private)
- una variabile di istanza **quantita** (tipo int, private)
- un costruttore che prende i 3 parametri sopra descritti;
- metodi pubblici accessori per tutti e 3 i parametri sopra descritti.

Si scriva una classe astratta **Ordine** con

- una variabile di istanza **cameriere** (tipo Personale, private) che rappresenta il cameriere che gestisce il tavolo;
- una variabile di istanza **comanda** (tipo ArrayList<ProdottoOrdinato>, final, private);

e i seguenti metodi di istanza:

- un costruttore che prende come parametro un **cameriere** e crea una comanda vuota associandola all'ordine;
- metodi pubblici accessori **`getCameriere()`**
- metodo pubblico astratto **`double getTotale()`** che restituisce l'importo totale dell'ordine.

La classe astratta **Ordine** è estesa dalle sottoclassi (non astratte) **OrdineTavolo** e **OrdineAsporto.**

Si tenga conto del fatto che la classe **OrdineTavolo** deve avere:

- una variabile di istanza **tavolo** (tipo String, private) che contiene l'identificativo del tavolo;
- una variabile di istanza **coperti** (tipo int, final, private) che contiene il numero di coperti.

e la classe **OrdineAsporto** deve avere:

- una variabile di istanza **nominativo** (tipo String, private) che contiene il nominativo della persona che ha effettuato l'ordine d'asporto.

Per entrambe le sottoclassi vanno scritti opportunamente i costruttori che richiamano il costruttore della classe **Ordine** e va implementato il metodo **`double getTotale(),`** tenendo conto del fatto che ogni coperto ha un costo di € 1,50 (nel caso di OrdineTavolo), mentre non c'è nessun costo aggiuntivo per gli ordini da asporto.

Si progetti infine una classe **FastFood** con

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

Implementare i seguenti metodi di istanza:

- un costruttore senza parametri che crea arraylist vuoti per persone e ordini.
- un metodo **public double totaleAsporto ()** che restituisce il totale degli importi degli ordini da asporto.
- un metodo **public Personale getMigliore ()** che restituisce il cameriere (tra quelli nell'arraylist persone) che ha servito tavoli per il più alto importo complessivo di ordini a loro associati (senza contare gli ordini da asporto). Si assuma che i camerieri associati agli ordini si riferiscano allo stesso oggetto cui si riferiscono i camerieri presnti nell'arraylist persone.

Esercizio 3 (5 punti)

Dato un intero positivo x ed un array a di n interi positivi maggiori si 1, progettare un algoritmo di programmazione dinamica che ritorni il valore **true** se e solo se esistono o meno indici i_1, i_2,…, i_k (non necessariamente distinti) nell'array a tali che:

$$\prod_{j=1}^{k} a[i_j] = x$$

Esempio: Si consideri a= {3, 7, 5, 2). È possibile ottenere 12 con tre indici 0,3 e 3 (ovvero a[0]*a[3]*a[3]=12).

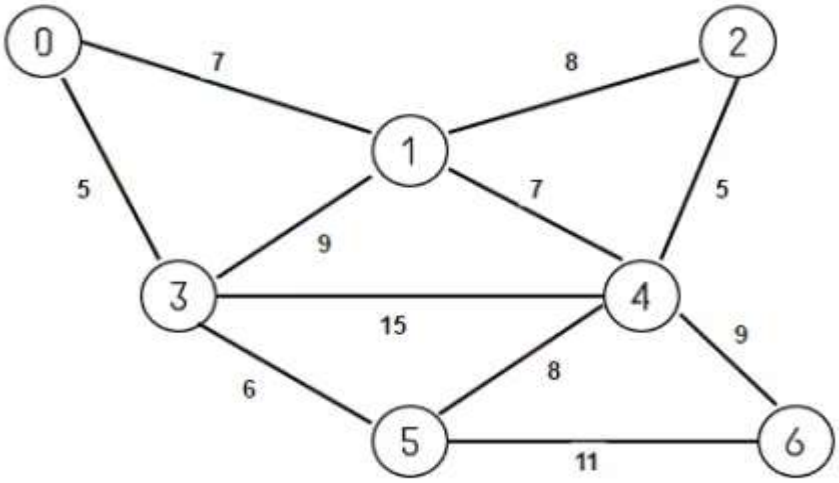
Suggerimento: indicare con F(i, x) la soluzione al problema che richiede di formare l'intero x utilizzando i primi i elementi dell'array a.

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

Nella terza riga (caso ricorsivo) bisogna considerare due casi: il caso in cui l'i-esimo elemento di a (cioè a[i-1]) concorre (almeno una volta) a formare x, e il caso in cui non concorre; nella quarta riga (altro caso ricorsivo), poiché a[i-1] non è un divisore di x, c'è da analizzare solo il caso in cui si può ottenere x come prodotto dei precedenti elementi dell'array.
Si fornisca una implementazione ricorsiva in Java dell'algoritmo proposto.

Esercizio 4 (15 punti)

Si consideri il grafo non diretto in figura.



a. **[7 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Dijkstra** per i cammini minimi da singola sorgente, a partire dal nodo sorgente **4**. Ad ogni passo, bisogna mostrare il contenuto della coda con priorità utilizzata dall'algoritmo.
b. **[8 punti]** Mostrare una possibile esecuzione **passo-passo** dell'algoritmo di **Floyd-Warshall** per i cammini minimi tra tutte le coppie, calcolando ad ogni passo la matrice delle distanze e la matrice dei padri π. Si dica se, nel grafo considerato, esiste, dal punto di vista computazionale, un modo più efficiente per risolvere il problema del calcolo dei cammini minimi tra tutte le coppie.

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.