Studente (Cognome Nome): _____ Matricola:_____

Corso di Informatica
Corso di Laurea in Ingegneria Gestionale
a.a. 2006-07
Secondo Compitino – 21 Dicembre 2006

Si noti che le soluzioni ai quesiti saranno considerate valide solo se il materiale consegnato includera' anche lo svolgimento. Tale foglio deve essere consegnato insieme allo svolgimento.

Quesito 1

Una magazzino di autoricambi è rappresentata da un array di ricambi, identificati dal nome del pezzo e dalla quantità di tali oggetti presenti in magazzino.

Un ordine effettuato da un cliente è rappresentato da un array di pezzi, identificati dal nome e dalla quantità ordinata.

a) Scrivere un metodo Java che verifichi che un ordine sia valido. Il metodo deve avere il seguente prototipo:

```
public static boolean is_valid(String [] mag_pezzo, int []
mag_quantità, String [] ordine_pezzo, int [] ordine_quantità)
```

e deve restituire true se:

1) tutti i ricambi richiesti nell'ordine sono presenti in magazzino

2) la quantità per ciascun elemento dell'ordine è <= della disponibilità in magazzino.

Prerequisiti del metodo: gli array non sono null e le quantità sono >= 0.

b) Scrivere un metodo Java che aggiorna l'elenco dei ricambi presenti in magazzino in relazione all'ordine ricevuto, decrementando il numero dei pezzi disponibili della quantità richiesta. Il metodo deve avere il seguente prototipo:

```
public static double agg_magazzino(String [] mag_pezzo, int []
mag_quantità, String [] ordine_pezzo, int [] ordine_quantità)
```

Prerequisiti del metodo: gli array non sono null e l'ordine è valido.

c) Scrivere un metodo Java che calcola il numero di elementi non validi all'interno di un ordine. Il metodo deve avere il seguente prototipo:

```
public static int non_validi(String [] mag_pezzo, int [] mag_quantità,
String [] ordine_pezzo, int [] ordine_quantità)
```

Prerequisiti del metodo: gli array non sono null.

d) Scrivere un programma main di test che

- Inizializza il magazzino con i seguenti valori:

Pezzo	Disponibilità
Cinghia lung.53	4
Freno diam.20	2
Pistone	7
Candela	20

- Crea un ordine non valido

- Se l’ordine non è valido, crei da questo, un ordine valido eliminando gli elementi che lo rendono non valido. Suggerimento: Quanti sono gli elementi presenti nel nuovo ordine (che deve essere valido)?

  Esempio: se l’ordine è :

Titolo	Quantità
Pistone	10
Antenna	1
Freno diam.20	1

  L’ordine valido derivato da questo è

Titolo	Quantità
Freno diam.20	1

- Aggiorna l’elenco dei pezzi di ricambio presenti in magazzino
- Stampa il nuovo elenco con un formato analogo alla tabella di sopra

Note:

- E’ possibile risolvere l’esercizio anche usando le classi. In tal caso, modificare in modo coerente l’intestazione delle funzioni.
- E’ possibile scrivere funzioni aggiuntive per risolvere dei compiti richiesti in più punti del programma.

Quesito 2

Si rappresenti il diagramma di flusso relativo al seguente metodo Java:

```
public static void f(int [] v) {
            int i=0;
            int N= v.length;
            while (i<N) {
                  v[i] +=i;
                  if (v[i]>10) {
                        v = new int[N];
                  }
                  i++;

            }
            System.out.println(v[N-1]);
      }
```

Quesito 3

Si determini l'output prodotto dall’esecuzione del seguente programma Java, in cui il corpo del metodo f e' stato omesso per brevita', essendo identico al metodo del quesito precedente.

```
public static void main(String[] args) {
            int [] a = new int [] {1,-15,18,23};
            System.out.println(a[1]+" "+a[2]+" "+a[3]);
            f(a);
            System.out.println(a[1]+" "+a[2]+" "+a[3]);
      }
```

Quesito 1: soluzione con metodi di supporto.

Note: il quesito può essere risolto in svariati modi. Qui viene proposto una soluzione che fa un uso estensivo dei metodi. Per un'altra soluzione che ricorre meno estensivamente ai metodi di supporto, vedi la soluzione alternativa proposta di seguito.

```
/**
 * Magazzino con ordini cliente
 *
 * @author Nicola Serreli
 *
 */
public class TestoC {
	/*
	 * Cerca un Pezzo (chiave) in magazzino, e restituisce l'indice se e' stato
	 * trovato, -1 altrimenti.
	 */
	public static int trova(String[] nomi, String chiave) {
		for (int i=0; i<nomi.length; i++) {
			if (nomi[i].equals(chiave)) {
				return i;
			}
		}
		return -1;
	}

	/*
	 * Controlla se un elemento (Nome + quantita') e' valido
	 */
	public static boolean valido(String[] nomi, int[] quantita, String chiave, int richiesti) {

		boolean valido = true;

		// controllo se il pezzo esiste
		int indice = trova(nomi, chiave);
		if (indice >-1) {
			// controllo la quantita'
			if (quantita[indice]<richiesti) {
				valido = false; // non valido
			}

		} else {
			valido = false; // non valido
```

```
		}

		return valido;
	}

	public static boolean is_valid(String[] mag_pezzo, int[] mag_quantita, String[] ordine_pezzo, int[] ordine_quantita) {
		// potrei controllare che tutti i pezzi soddisfino la condizione
		// oppure semplicemente controllare che nessun pezzo sia "non valido" utilizzando al funzione non_validi

		return non_validi(mag_pezzo, mag_quantita, ordine_pezzo, ordine_quantita) == 0;
	}

	public static int non_validi(String[] mag_pezzo, int[] mag_quantita, String[] ordine_pezzo, int[] ordine_quantita) {
		int numero = 0;
		for (int i=0; i<ordine_pezzo.length; i++) {

			// controllo se il pezzo esiste e se la quantita' e' quella giusta
			boolean valido = valido(mag_pezzo, mag_quantita, ordine_pezzo[i], ordine_quantita[i]);
			if (!valido) {
				numero++; // non valido
			}
		}
		return numero;
	}

	public static void aggiorna(String[] mag_pezzo, int[] mag_quantita, String[] ordine_pezzo, int[] ordine_quantita) {

		for (int i=0; i<ordine_pezzo.length; i++) {

			// cerco il pezzo (presuppongo che esista)
			int indice = trova(mag_pezzo, ordine_pezzo[i]);

			// aggiorno la quantita'
			mag_quantita[indice] -= ordine_quantita[i];
		}
	}

	/*
	 * Stampa il catalogo
```

```
	 */
	public static void stampa(String[] mag_pezzo, int[] mag_quantita) {
		System.out.println("Pezzo\tQuantità");
		for (int i=0; i<mag_pezzo.length; i++) {

			System.out.println(" "+ mag_pezzo[i] + "\t"+ mag_quantita[i]);
		}
	}


	public static void main(String[] args) {
		/*
		 *
		 *  Definizione ed inizializzazione magazzino ed Ordine
		 *
		 */
		String[] mag_pezzo = new String[] {"Cinghia lung.53", "Freno diam.20", "Pistone", "Candela"};
		int[] mag_quantita = new int[] {4, 2, 7, 20};
		String[] ordine_pezzo = new String[] {"Pistone", "Antenna", "Freno diam.20"};
		int[] ordine_quantita = new int[] {10, 1, 1};

		/*
		 *
		 *  controllo se l'ordine e' valido
		 *
		 */
		if (!is_valid(mag_pezzo, mag_quantita, ordine_pezzo, ordine_quantita)) {
			/*
			 *
			 *  creo un nuovo ordine (valido)
			 *
			 */
			int tot_non_validi = non_validi(mag_pezzo, mag_quantita, ordine_pezzo, ordine_quantita);
			String[] tmp_pezzo = new String[ordine_pezzo.length - tot_non_validi];
			int[] tmp_quantita  = new int[ordine_pezzo.length - tot_non_validi];

			int i;   // percorre l'ordine vecchio
			int j=0; // percorre l'ordine nuovo

			for (i=0; i<ordine_pezzo.length; i++) {
				// controllo se il pezzo esiste e se la quantita' e' quella giusta
				boolean valido = valido(mag_pezzo, mag_quantita, ordine_pezzo[i],
ordine_quantita[i]);
```

```
                        if (valido) {
                                // lo salvo
                                tmp_pezzo[j]   = ordine_pezzo[i];
                                tmp_quantita[j] = ordine_quantita[i];

                                j++; // mi ricordo che ho salvato un elemento nel nuovo ordine
                        }
                }

                ordine_pezzo = tmp_pezzo;
                ordine_quantita = tmp_quantita;
        }
        /*
         *
         * Aggiorno il catalogo
         *
         */
        aggiorna(mag_pezzo, mag_quantita, ordine_pezzo, ordine_quantita);

        /*
         *
         * Stampa il catalogo
         *
         */
        stampa(mag_pezzo, mag_quantita);
    }
}
```

Quesito 1: soluzione Alternativa.

Note:questa soluzione è simile alla via proposta per il quesito A.

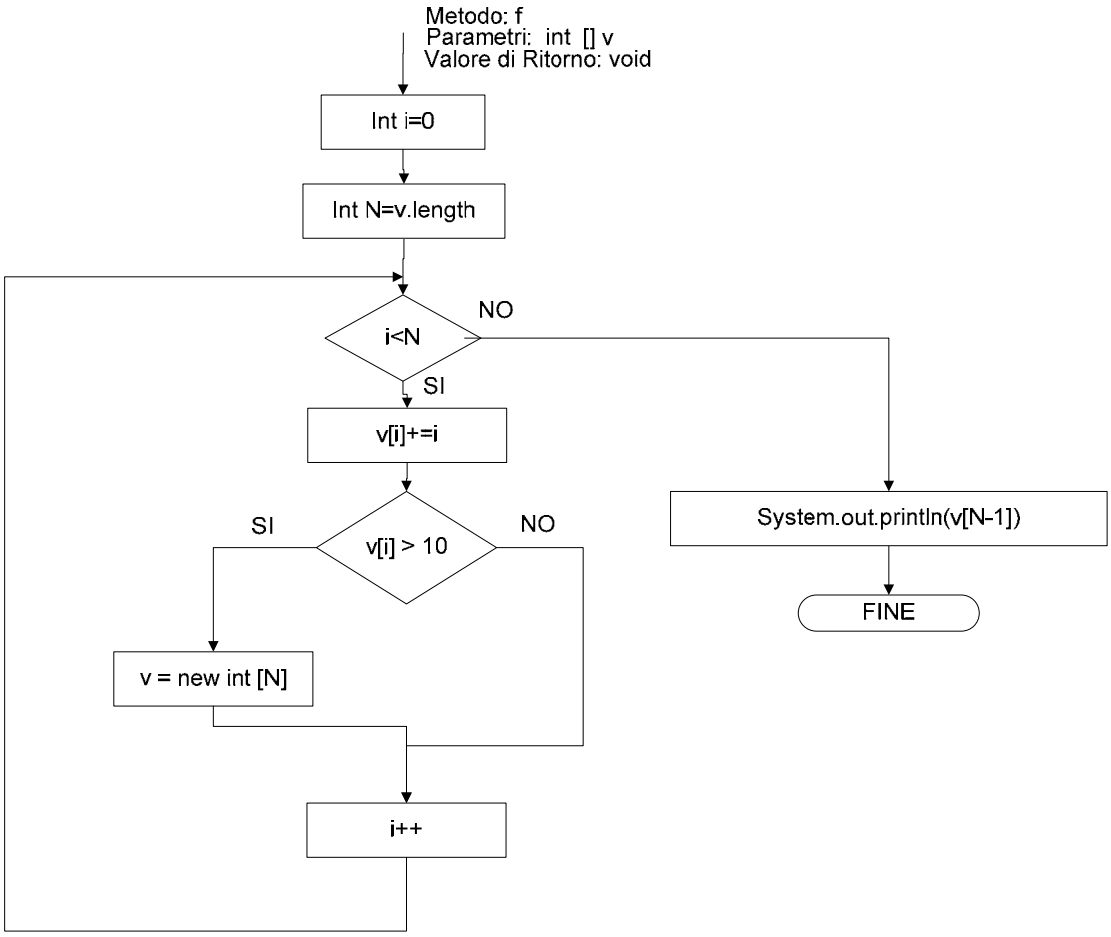
```
public class testoC {

      public static boolean is_valid(
            String[] mag_pezzo,
            int[] mag_quantita,
            String[] ordine_pezzo,
            int[] ordine_quantita) {
            // prima soluzione: si contano i numeri degli elementi dell'ordine
            // che sono validi.
            // se sono uguali alla lunghezza dell'ordine, l'ordine è valido
            boolean valido = false; // l'inizializzazione non è imp.
            int num_validi = 0; // numero di ordini validi
            int i; // indice per ciclare sugli ordini
            int j; // indice per ciclare sui pezzi in magazzino
            for (i = 0; i < ordine_pezzo.length; i++) {
```

```
                for (j = 0; j < mag_pezzo.length; j++) {
                      // cerca un pezzo dell'ordine
                      // presente in magazzino
                      if (ordine_pezzo[i].equals(mag_pezzo[j]))
                            // condizione di validita
                            // nota:non testo il caso di mag_quantita<0
                            // perchè è dato come prerequisito
                            if (ordine_quantita[i] <= mag_quantita[j])
                                  num_validi++; //la riga dell'ordine è valida

                } // end_for_j
          } // end_for_i

          // test di validità:
          // il numeo di righe valide deve
          // essere uguale alla lunghezza dell'ordine
          if (num_validi == ordine_pezzo.length)
                valido = true;
          else
                valido = false;

          return (valido);
    } //end_is_valid

    // seconda soluzione
    // un ordine è valido
    // se il numero di entry non valide è pari a 0
    public static boolean is_valid1(
          String[] mag_pezzo,
          int[] mag_quantita,
          String[] ordine_pezzo,
          int[] ordine_quantita) {

          boolean valido;
          if (non_validi(mag_pezzo, mag_quantita, ordine_pezzo,
ordine_quantita)
                == 0)
                valido = true;
          else
                valido = false;

          return valido;

    }

    public static void aggiorna(
          String[] mag_pezzo,
          int[] mag_quantita,
          String[] ordine_pezzo,
          int[] ordine_quantita) {
          // per ogni pezzo ordinato,
          // lo cerco in magazzino
          // ed aggiorno le relative quantità
          //nota:non occorrono controlli
          // perchè l'ordine è valido
          int i; // contatore sugli ordini
          int j; // contatore sui pezzi in magazzino
          for (i = 0; i < ordine_pezzo.length; i++) {
                for (j = 0; j < mag_pezzo.length; j++) {
                      if (ordine_pezzo[i].equals(mag_pezzo[j]))
                            // aggiorno la quantita'
                            mag_quantita[j] -= ordine_quantita[i];
                } // end_for_j
          } // end_for_i
```

```
	} //end_aggiorna

	public static int non_validi(
		String[] mag_pezzo,
		int[] mag_quantita,
		String[] ordine_pezzo,
		int[] ordine_quantita) {
		// si contano i numeri degli elementi dell'ordine
		// che sono validi.
		// il numero di elementi non validi è la lunghezza dell'array
		// degli ordini - il numero di entry non valide
		int non_validi = 0; // numero di righe non valide
		int num_validi = 0; // numero di ordini validi
		int i; // indice per ciclare sugli ordini
		int j; // indice per ciclare sui pezzi in magazzino
		for (i = 0; i < ordine_pezzo.length; i++) {
			for (j = 0; j < mag_pezzo.length; j++) {
				// cerca un pezzo dell'ordine
				// presente in magazzino
				if (ordine_pezzo[i].equals(mag_pezzo[j]))
					// condizione di validita
					if ((ordine_quantita[i] >= 0)
						&& (ordine_quantita[i] <= mag_quantita[j]))
						num_validi++; //la riga dell'ordine è valida

			} // end_for_j
		} // end_for_i

		non_validi = ordine_pezzo.length - num_validi;

		return (non_validi);
	} //end_non_validi

	public static void main(String[] args) {

		// inizializzazione
		String[] mag_pezzo =
			new String[] {
				"Cinghia lung.53",
				"Freno diam.20",
				"Pistone",
				"Candela" };
		int[] mag_quantita = new int[] { 4, 2, 7, 20 };
		//    creazione ordine non valido
		// (quello suggerito nel testo)
		String[] ordine_pezzo =
			new String[] { "Pistone", "Antenna", "Freno diam.20" };
		int[] ordine_quantita = new int[] { 10, 1, 1 };

		boolean valido =
			is_valid(mag_pezzo, mag_quantita, ordine_pezzo,
ordine_quantita);

		if (!valido) {
			// creo un ordine temporaneo fatto da due array paralleli.
			// la dimensione è pari alla dimensione del vecchio
			// ordine - il numero di entry non valide
			int elem_non_validi =
				non_validi(
					mag_pezzo,
					mag_quantita,
					ordine_pezzo,
					ordine_quantita);
			int lunghezza = ordine_pezzo.length - elem_non_validi;
```

```
                  String[] tmp_ord_pezzo = new String[lunghezza];
                  int[] tmp_ord_quantita = new int[lunghezza];

                  // tutti gli elementi validi li ricopio negli array fittizi
                  int i, j; // due indici per scorrere il magazzino e gli ordini
                  int k = 0; // indice dell'elemento inserito nell'array
temporaneo
                  for (i = 0; i < ordine_pezzo.length; i++)
                        for (j = 0; j < mag_pezzo.length; j++) {
                              // condizioni di validita di una entry
                              if (ordine_pezzo[i].equals(mag_pezzo[j])) {
                                    if ((ordine_quantita[i] >= 0)
                                          && (ordine_quantita[i] <=
mag_quantita[j])) {
                                          // l'entry è valida, la ricopio
nell'ordine
                                          // temporaneo
                                          tmp_ord_pezzo[k] = ordine_pezzo[i];
                                          tmp_ord_quantita[k] =
ordine_quantita[i];
                                          k++;
                                    } // end_if ordine_quantita
                              } // end_if ordine_titolo
                        } // end_for_j

                  // ora butto via il vecchio ordine e prendo l'ordine valido
                  ordine_pezzo = tmp_ord_pezzo;
                  ordine_quantita = tmp_ord_quantita;

            } // end_if_not_valido

            // aggiorno il magazzino
            aggiorna(mag_pezzo, mag_quantita, ordine_pezzo, ordine_quantita);

            // stampo il magazzino
            System.out.println(" pezzo\tQuantità");
            for (int i = 0; i < mag_pezzo.length; i++) {
                  System.out.println(" " + mag_pezzo[i] + "\t" +
mag_quantita[i]);
            }

      }

}
```

Quesito 2: soluzione.



Quesito 3: soluzione.

L'output del programma è il seguente:

```
-15 18 23
3
-14 20 23
```

Per la giustificazione, si ricorre alla evoluzione dello stato del programma, identificando i contenuti dell'area locale ed area globale e seguendo la loro evoluzione. Attenzione, nel metodo, a quando viene costruito il nuovo oggetto.