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 preventivo inviato da un fornitore è rappresentato da un array di pezzi, identificati dal nome e dalla quantità proposta.

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

```
public static boolean is_valid(String [] mag_pezzo, int []
mag_quantità, String [] preventivo_pezzo, int [] preventivo_quantità)
```

e deve restituire true se:

1) tutti i ricambi presenti nel preventivo sono presenti in magazzino

2) la quantità per ciascun elemento del preventivo è >0 e tale che sommata alla quantità disponibile in magazzino dia un numero <=100 (ossia, in magazzino, dopo aver effettuato l'acquisto, per ogni pezzo non possono essere presenti più di 100 elementi). 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'accettazione del preventivo ricevuto, incrementando il numero dei pezzi disponibili della quantità proposta. Il metodo deve avere il seguente prototipo:

```
public static double agg_magazzino(String [] mag_pezzo, int []
mag_quantità, String [] preventivo_pezzo, int [] preventivo_quantità)
```

Prerequisiti del metodo: gli array non sono null e il preventivo è valido.

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

```
public static int non_validi(String [] mag_pezzo, int [] mag_quantità,
String [] preventivo_pezzo, int [] preventivo_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 preventivo non valido
- Se il preventivo non è valido, crei da questo un preventivo valido eliminando gli elementi che lo rendono non valido. Suggerimento: Quanti sono gli elementi presenti nel nuovo preventivo (che deve essere valido)?

  Esempio: se il preventivo è :

Titolo	Quantità
Pistone	100
Antenna	1
Freno diam.20	1

  Il preventivo 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;
            do {
                  if ((i % 3) == 0) {
                        v[i]++;
                  }

                  i++;
            } while (i < N);
      }
```

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, 1, 4, 3 };
            System.out.println(a[0] + " " + a[1] + " " + a[2] + " " + a[3]);
            f(a);
            System.out.println(a[0] + " " + 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 preventivo Fornitore
 *
 * @author Nicola Serreli
 *
 */
public class TestoD {

	/*
	 * 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>100) {
				valido = false; // non valido
			}
```

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

		return valido;
	}

	public static boolean is_valid(String[] mag_pezzo, int[] mag_quantita, String[] preventivo_pezzo, int[] preventivo_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, preventivo_pezzo, preventivo_quantita) == 0;
	}

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

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

	public static void aggiorna(String[] mag_pezzo, int[] mag_quantita, String[] preventivo_pezzo, int[] preventivo_quantita) {

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

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

			// aggiorno la quantita'
			mag_quantita[indice] += preventivo_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 il preventivo
		 *
		 */
		String[] mag_pezzo = new String[] {"Cinghia lung.53", "Freno
diam.20", "Pistone", "Candela"};
		int[] mag_quantita = new int[] {4, 2, 7, 20};

		//    creazione preventivo non valido
		// (quello suggerito nel testo)
		String[] preventivo_pezzo = new String[] {"Pistone", "Antenna",
"Freno diam.20"};
		int[] preventivo_quantita = new int[] {100, 1, 1};
		/*
		 *
		 *  controllo se l'preventivo e' valido
		 *
		 */
		if (!is_valid(mag_pezzo, mag_quantita, preventivo_pezzo, preventivo_quantita)) {
			/*
			 *
			 *  creo un nuovo preventivo (valido)
			 *
			 */
			int tot_non_validi = non_validi(mag_pezzo, mag_quantita, preventivo_pezzo,
preventivo_quantita);
			String[] tmp_pezzo = new String[preventivo_pezzo.length - tot_non_validi];
			int[] tmp_quantita  = new int[preventivo_pezzo.length - tot_non_validi];
```

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

                for (i=0; i<preventivo_pezzo.length; i++) {
                        // controllo se il pezzo esiste e se la quantita' e' quella giusta
                        boolean valido = valido(mag_pezzo, mag_quantita, preventivo_pezzo[i], preventivo_quantita[i]);
                        if (valido) {
                                // lo salvo
                                tmp_pezzo[j]   = preventivo_pezzo[i];
                                tmp_quantita[j] = preventivo_quantita[i];

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

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

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

Quesito 1: soluzione Alternativa.

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

```
public class testoD {

      public static boolean is_valid(
            String[] mag_pezzo,
            int[] mag_quantita,
            String[] preventivo_pezzo,
            int[] preventivo_quantita) {
```

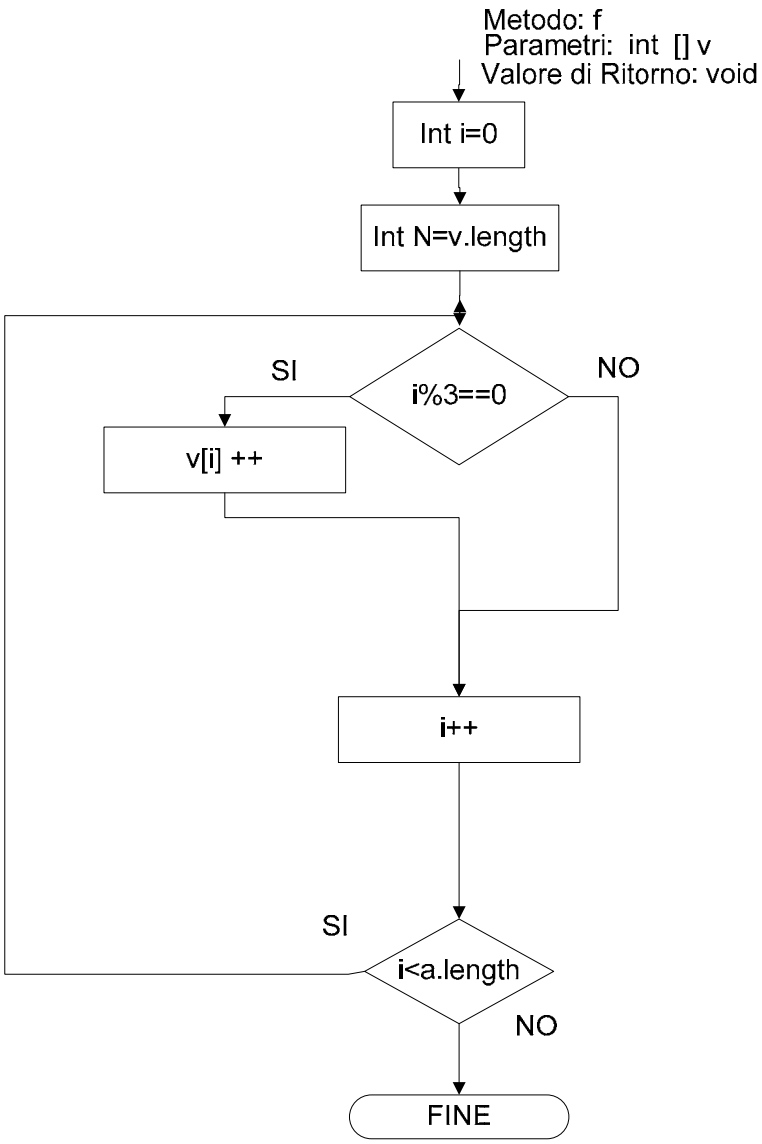
```
            // prima soluzione: si contano i numeri degli elementi del
preventivo
            // che sono validi.
            // se il numero è uguale alla lunghezza del preventivo, il
preventivo è
            // valido
            boolean valido = false; // l'inizializzazione non è imp.
            int num_validi = 0; // numero di preventivi validi
            int i; // indice per ciclare sui preventivi
            int j; // indice per ciclare nel magazzino
            for (i = 0; i < preventivo_pezzo.length; i++) {
                  for (j = 0; j < mag_pezzo.length; j++) {
                        // cerca un titolo del preventivo
                        // presente in magazzino
                        if (preventivo_pezzo[i].equals(mag_pezzo[j]))
                              // condizione di validita
                              // nota:non testo il caso di preventivo_quantita<0
                              // perchè è dato come prerequisito
                              if (preventivo_quantita[i] + mag_quantita[j] <=
100)
                                    num_validi++; //la riga del preventivo è
valida

                  } // end_for_j
            } // end_for_i

            // test di validità:
            // il numeo di righe valide deve
            // essere uguale alla lunghezza del preventivo
            if (num_validi == preventivo_pezzo.length)
                  valido = true;
            else
                  valido = false;

            return (valido);
      } //end_is_valid

      // seconda soluzione suggerita durante l'esame
      // un preventivo 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[] preventivo_pezzo,
            int[] preventivo_quantita) {

            boolean valido;
            if (non_validi(mag_pezzo,
                  mag_quantita,
                  preventivo_pezzo,
                  preventivo_quantita)
                  == 0)
                  valido = true;
            else
                  valido = false;

            return valido;

      }

      public static void aggiorna(
            String[] mag_pezzo,
            int[] mag_quantita,
            String[] preventivo_pezzo,
```

```
            int[] preventivo_quantita) {
            // per ogni pezzo del preventivo,
            // lo cerco in magazzino
            // ed aggiorno le relative quantità
            //nota:non occorrono controlli
            // perchè il preventivo è valido
            int i; // contatore sui preventivi
            int j; // contatore sui pezzi in magazzino
            for (i = 0; i < preventivo_pezzo.length; i++) {
                  for (j = 0; j < mag_pezzo.length; j++) {
                        if (preventivo_pezzo[i].equals(mag_pezzo[j]))
                              // aggiorno la quantita'
                              mag_quantita[j] += preventivo_quantita[i];
                  } // end_for_j
            } // end_for_i
      } //end_aggiorna

      public static int non_validi(
            String[] mag_pezzo,
            int[] mag_quantita,
            String[] preventivo_pezzo,
            int[] preventivo_quantita) {
            // si contano i numeri degli elementi del preventivo
            // che sono validi.
            // il numero di elementi non validi è la lunghezza dell'array
            // dei preventivi - il numero di entry non valide
            int non_validi = 0; // numero di righe non valide
            int num_validi = 0; // numero di preventivi validi
            int i; // indice per ciclare sui preventivi
            int j; // indice per ciclare nel magazzino
            for (i = 0; i < preventivo_pezzo.length; i++) {
                  for (j = 0; j < mag_pezzo.length; j++) {
                        // cerca un pezzo del preventivo
                        // presente in magazzino
                        if (preventivo_pezzo[i].equals(mag_pezzo[j]))
                              // condizione di validita
                              if ((preventivo_quantita[i] >= 0)
                                    && (preventivo_quantita[i] + mag_quantita[j]
<= 100))
                                    num_validi++; //la riga del preventivo è
valida

                  } // end_for_j
            } // end_for_i

            non_validi = preventivo_pezzo.length - num_validi;

            return (non_validi);
      } //end_non_validi

      public static void main(String[] args) {

            // inizializzazione magazzino
            String[] mag_pezzo = new String[] {"Cinghia lung.53", "Freno
diam.20", "Pistone", "Candela"};
            int[] mag_quantita = new int[] {4, 2, 7, 20};

            //    creazione preventivo non valido
            // (quello suggerito nel testo)
            String[] preventivo_pezzo = new String[] {"Pistone", "Antenna",
"Freno diam.20"};
            int[] preventivo_quantita = new int[] {100, 1, 1};

            boolean valido =
```

```
                  is_valid(
                        mag_pezzo,
                        mag_quantita,
                        preventivo_pezzo,
                        preventivo_quantita);

            if (!valido) {
                  // creo un preventivo temporaneo fatto da due array paralleli.
                  // la dimensione è pari alla dimensione del vecchio
                  // preventivo - il numero di entry non valide
                  int elem_non_validi =
                        non_validi(
                              mag_pezzo,
                              mag_quantita,
                              preventivo_pezzo,
                              preventivo_quantita);
                  int lunghezza = preventivo_pezzo.length - elem_non_validi;
                  String[] tmp_prev_pezzo = new String[lunghezza];
                  int[] tmp_prev_quantita = new int[lunghezza];

                  // tutti gli elementi validi li ricopio negli array fittizi
                  int i, j; // due indici per scorrere il magazzino e i
preventivi
                  int k = 0; // indice dell'elemento inserito nell'array
temporaneo
                  for (i = 0; i < preventivo_pezzo.length; i++)
                        for (j = 0; j < mag_pezzo.length; j++) {
                              // condizioni di validita di una entry
                              if (preventivo_pezzo[i].equals(mag_pezzo[j])) {
                                    if ((preventivo_quantita[i] >= 0)
                                          && (preventivo_quantita[i] +
mag_quantita[j]
                                                <= 100)) {
                                          // l'entry è valida, la ricopio nel
preventivo
                                          // temporaneo
                                          tmp_prev_pezzo[k] =
preventivo_pezzo[i];
                                          tmp_prev_quantita[k] =
preventivo_quantita[i];
                                          k++;
                                    } // end_if preventivo_quantita
                              } // end_if preventivo_titolo
                        } // end_for_j

                  // ora butto via il vecchio preventivo e prendo il preventivo
                  // valido
                  preventivo_pezzo = tmp_prev_pezzo;
                  preventivo_quantita = tmp_prev_quantita;

            } // end_if_not_valido

            // aggiorno il magazzino
            aggiorna(
                  mag_pezzo,
                  mag_quantita,
                  preventivo_pezzo,
                  preventivo_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:

```
1 1 4 3
2 1 4 4
```

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, ai valori del modulo.