Studente (Cognome Nome): _____ Matricola:_____

Corso di Informatica
Corso di Laurea in Ingegneria Gestionale
a.a. 2006-07
Primo Scritto – 12 Gennaio 2007

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 libreria è rappresentata da un array di libri, identificati dal titolo e dal prezzo di ciascun titolo. Un preventivo richiesto da un cliente è rappresentato da un array di libri, identificati dal titolo e dalla quantità richiesta.

a) Scrivere un metodo Java che calcoli il costo di un preventivo. Il metodo deve avere il seguente prototipo:

```
public static double costo(String [] lib_titolo, double [] lib_prezzo,
String [] prev_titolo, int [] prev_quantità)
```

Prerequisiti del metodo: gli array non sono null, i prezzi e le quantità sono >=0, tutti i titoli del preventivo sono presenti in libreria.

b) Scrivere un metodo Java che ordini in modo crescente un preventivo rispetto al costo dei singoli elementi (costo=quantità X prezzo).. Il metodo deve avere il seguente prototipo:

```
public static void ordina(String [] lib_titolo, double [] lib_prezzo,
String [] prev_titolo, int [] prev_quantità)
```

Prerequisiti del metodo: gli array non sono null, i prezzi e le quantità sono >=0, tutti i titoli del preventivo sono presenti in libreria. Esempio:

se il contenuto della libreria è il seguente,

Titolo	prezzo
Analisi I	100
C++	220
Visual Basic	150
Visio	300

e il preventivo è il seguente

Titolo	Quantità
C++	5
Visual Basic	1
Analisi I	6

L'invocazione del metodo di cui sopra riorganizza il preventivo nel seguente modo:

Titolo	Quantità
Visual Basic	1
Analisi I	6
C++	5

c) Scrivere un metodo Java che individua gli elementi non validi all'interno di un preventivo. Il metodo deve avere il seguente prototipo (il preventivo si suppone ordinato in modo crescente rispetto al costo degli elementi, come indicato dal metodo b):

```
public static boolean [] validi(String [] lib_titolo, double []
lib_prezzo, String [] prev_titolo, int [] prev_quantità)
```

un preventivo è valido se il costo del preventivo è <= 1000 Euro. Se il preventivo è valido, il metodo deve restituire un vettore le cui componenti sono tutte true. Se il preventivo non è valido, ossia il costo è > 1000 Euro, sono considerati validi gli elementi a costo minore, (costo = prezzoXquantità), la cui somma dei costi è <= 1000 Euro.

Esempio: se il contenuto della libreria è il seguente:

Titolo	prezzo
Analisi I	100
C++	220
Visual Basic	150
Visio	300

E l'ordine è il seguente:

Titolo	Quantità
Visual Basic	1
Analisi I	6
C++	5

Il metodo restituisce l'array {true, true, false}.

Prerequisiti del metodo: gli array non sono null, i prezzi e le quantità sono >=0, tutti i titoli del preventivo sono presenti in libreria, il preventivo si suppone ordinato in modo crescente rispetto al costo degli elementi, come indicato dal metodo b

Quesito 2

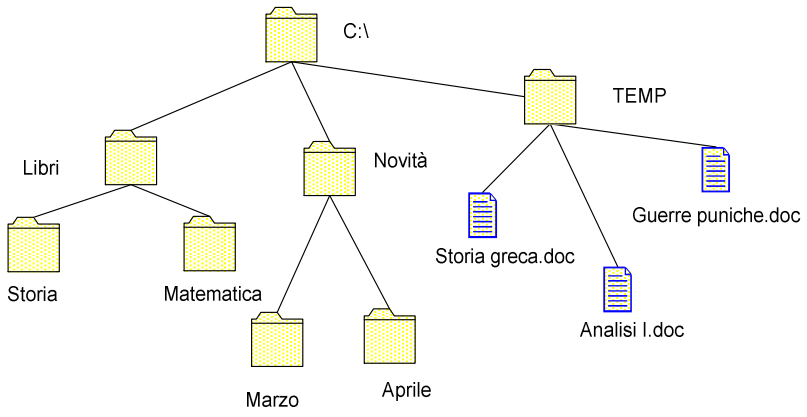
a) Rappresentare in C2 su 8 bit i seguenti numeri:

```
-28
+126
```

b) Calcolare la somma di tali numeri su 8 bit se rappresentabile.

Quesito 3

Si consideri il seguente file system, di cui esistono solo la directory C:\ e TEMP, con i relativi file contenuti (non directory). Si tenga presente che nel mese di marzo è arrivato "Analisi I.doc", nel mese di aprile "Storia greca.doc".



a) Impartire i comandi per creare le directory mancanti, supponendo che la directory corrente sia C:\.

b) Impartire la sequenza di comandi per spostare i file da C:\TEMP nelle relative cartelle/directory, eventualmente creando link, utilizzando solo path-name relativi. Si suppone che la directory corrente sia C:\TEMP. É possibile navigare fra le directory utilizzando il comando cd.

Soluzione della parte Java

Vengono proposte due soluzioni, la prima senza il ricorso alle classi, la seconda con l'utilizzo di classi.

Soluzione senza classi:

```
/**
 * Soluzione scritto del 12 gennaio 07
 *
 * Libreria
 *
 * @author Nicola Serreli
 */
public class PrimoScritto_A {

	public static double costo(String[] lib_titolo, double[] lib_prezzo,
			String[] prev_titolo, int[] prev_quantita) {

		double costo_totale = 0;

		int i = 0;
		int j = 0;
		for (i = 0; i < prev_titolo.length; i++) {
			/*
			 * Per ogni libro in preventivo :
			 * 1) lo cerco in libreria.
			 * 2) conoscendo il prezzo, calcolo il suo costo.
			 */

			for (j = 0; j < lib_titolo.length; j++) {
				if (prev_titolo[i].equals(lib_titolo[j])) {
					// i due elementi si riferiscono allo stesso libro, quindi
					// posso calcolare il costo del libro
					double costo_libro = prev_quantita[i] * lib_prezzo[j];
					costo_totale += costo_libro;
				}
			}

		}
		return costo_totale;
	}

	public static void ordina(String[] lib_titolo, double[] lib_prezzo,
			String[] prev_titolo, int[] prev_quantita) {
		int i = 0;
		int j = 0;

		/*
		 * Per ordinare il preventivo, e' utile avere un vettore contenente i
		 * costi dei singoli libri.
		 *
		 * Notare che questo vettore e' "parallelo" ai vettori del preventivo,
		 * quindi nell'ordinare dovro' fare 3 scambi.
		 */
		double[] prev_costi = new double[prev_titolo.length];

		/* calcolo i vari costi (simile alla funzione costo) */
		for (i = 0; i < prev_titolo.length; i++) {

			for (j = 0; j < lib_titolo.length; j++) {
				if (prev_titolo[i].equals(lib_titolo[j])) {
					// Nota: l'indice dei costi e' lo stesso del preventivo
					prev_costi[i] = prev_quantita[i] * lib_prezzo[j];
				}
			}
		}

		/* ordino rispetto al nuovo vettore */
		for (i = 0; i < (prev_costi.length - 1); i++) {

			// cerco il minimo
			double minimo = prev_costi[i];
			int posizione = i;

			for (j = i + 1; j < prev_costi.length; j++) {
```

```
                        if (minimo > prev_costi[j]) {
                                // Aggiorno il minimo
                                minimo = prev_costi[j];
                                posizione = j;
                        }
                }

                // scambi
                double tmp_c = prev_costi[i];
                prev_costi[i] = prev_costi[posizione];
                prev_costi[posizione] = tmp_c;

                String tmp_t = prev_titolo[i];
                prev_titolo[i] = prev_titolo[posizione];
                prev_titolo[posizione] = tmp_t;

                int tmp_i = prev_quantita[i];
                prev_quantita[i] = prev_quantita[posizione];
                prev_quantita[posizione] = tmp_i;
        }
}


public static boolean[] validi(String[] lib_titolo, double[] lib_prezzo,
                String[] prev_titolo, int[] prev_quantita) {

        /*
         * Per risolvere questo quesito ho bisogno di conoscere il costo dei
         * singoli elementi e di sommare gli elementi a costo minore, finche' il
         * costo totale rimane <= 1000. Poiche' l'ordine e' crescente, gli
         * elementi a costo minore stanno nelle prime posizioni.
         */

        boolean[] prev_validi = new boolean[prev_titolo.length];
        double costo_totale = 0;

        int i = 0;
        int j = 0;
        for (i = 0; i < prev_titolo.length; i++) {
                /*
                 * Per ogni libro in preventivo: lo cerco in libreria e ne calcolo
                 * il costo
                 */

                for (j = 0; j < lib_titolo.length; j++) {
                        if (prev_titolo[i].equals(lib_titolo[j])) {
                                // i due elementi si riferiscono allo stesso libro, quindi
                                // posso calcolare il costo del libro
                                double costo_libro = prev_quantita[i] * lib_prezzo[j];
                                costo_totale += costo_libro;

                                // controllo se l'elemento e' valido, cioe' se non fa salire
                                // il costo totale oltre i 1000
                                if (costo_totale <= 1000) {
                                        prev_validi[i] = true;
                                } else {
                                        prev_validi[i] = false;
                                }
                        }
                }

        }

        return prev_validi;
}

// soluzione alternativa
public static boolean[] validi_2(String[] lib_titolo, double[] lib_prezzo,
                String[] prev_titolo, int[] prev_quantita) {

        int i, j;

        /*
         * Per risolvere questo quesito ho bisogno di conoscere il costo dei
         * singoli elementi e di sommare gli elementi a costo minore, finche' il
         * costo totale rimane <= 1000. Poiche' l'ordine e' crescente, gli
         * elementi a costo minore stanno nelle prime posizioni.
         */

        boolean[] prev_validi = new boolean[prev_titolo.length];
```

```
		double[] prev_costi = new double[prev_titolo.length];

		/* calcolo i vari costi (simile alla funzione costo) */
		for (i = 0; i < prev_titolo.length; i++) {

			for (j = 0; j < lib_titolo.length; j++) {
				if (prev_titolo[i].equals(lib_titolo[j])) {
					// Nota: l'indice dei costi e' lo stesso del preventivo
					prev_costi[i] = prev_quantita[i] * lib_prezzo[j];
				}
			}
		}


		// il costo totale puo' essere calcolato con la funzione costo
		double costo_totale = costo(lib_titolo, lib_prezzo,
				prev_titolo, prev_quantita);


		// NOTA : inizio dall'elemento + caro (l'ultimo) e li sottraggo al
		// preventivo, finche' il totale non e' <= 1000
		i = prev_costi.length -1;
		while (costo_totale>1000) {
			costo_totale -= prev_costi[i];
			prev_validi[i] = false;
			i--;
		}

		// tutti gli altri elementi saranno validi
		for (; i >=0 ; i--) {
			prev_validi[i] = true;
		}

		return prev_validi;
	}

	// main di test, non richiesto nel compito
	//

	public static void main(String[] args) {
		String[] lib_titolo = new String[] {"Analisi I", "C++", "Visual Basic", "Visio"};
		double[] lib_prezzo = new double[] {100, 225, 150, 300};
		String[] prev_titolo = new String[] {"C++", "Visual Basic", "Analisi I"};
		int[] prev_quantita = new int[] {5, 1, 6};

		// costo
		double costo = costo(lib_titolo, lib_prezzo, prev_titolo, prev_quantita);
		System.out.println("Il costo totale e' " + costo);

		// ordinamento
		ordina(lib_titolo, lib_prezzo, prev_titolo, prev_quantita);
		System.out.println("Ordinato ");
		for (int i=0; i<prev_titolo.length; i++) {
			System.out.println("  " + prev_titolo[i] + "\t"+prev_quantita[i]);
		}

		// validita'
		boolean[] valid_1 = validi(lib_titolo, lib_prezzo, prev_titolo, prev_quantita);
		System.out.print("Validita' (1) =");
		for (int i=0; i<valid_1.length; i++) {
			System.out.print("  " + valid_1[i]);
		}
		System.out.println();

		boolean[] valid_2 = validi(lib_titolo, lib_prezzo, prev_titolo, prev_quantita);
		System.out.print("Validita' (2) =");
		for (int i=0; i<valid_2.length; i++) {
			System.out.print("  " + valid_2[i]);
		}
		System.out.println();
	}

}
```

Soluzione con classi:

```
/**
 * Soluzione scritto del 12 gennaio 07
 *
 * Libreria con classi
 *
 * @author Nicola Serreli
 */

/** Contiene un elemento del preventivo */
class Prev_libro {
	private String nome;
	private int quantita;

	public Prev_libro(String nome, int quantita) {
		this.nome = nome;
		this.quantita = quantita;
	}

	/**
	 * Restituisce il nome.
	 */
	public String getNome() {
		return nome;
	}
	/**
	 * Restituisce la quantita.
	 */
	public int getQuantita() {
		return quantita;
	}
}

/** Contiene un elemento della libreria */
class Lib_libro {
	private String nome;
	private double prezzo;

	public Lib_libro(String nome, double prezzo) {
		this.nome = nome;
		this.prezzo = prezzo;
	}

	/**
	 * Restituisce il nome.
	 */
	public String getNome() {
		return nome;
	}
	/**
	 * Restituisce il prezzo.
	 */
	public double getPrezzo() {
		return prezzo;
	}
}

public class PrimoScritto_A_classi {

	public static double costo(Lib_libro[] libreria, Prev_libro[] preventivo) {

		double costo_totale = 0;

		int i = 0;
		int j = 0;
		for (i = 0; i < preventivo.length; i++) {
			/*
			 * Per ogni libro in preventivo :
			 * 1) lo cerco in libreria.
			 * 2) conoscendo il prezzo, calcolo il suo costo.
			 */
			String titolo = preventivo[i].getNome();

			for (j = 0; j < libreria.length; j++) {
				if (titolo.equals(libreria[j].getNome())) {
					// i due elementi si riferiscono allo stesso libro, quindi
					// posso calcolare il costo del libro
					double costo_libro = preventivo[i].getQuantita() *
libreria[j].getPrezzo();
					costo_totale += costo_libro;
				}
```

```
                }

        }
        return costo_totale;
    }

    public static void ordina(Lib_libro[] libreria, Prev_libro[] preventivo) {
        int i = 0;
        int j = 0;

        /*
         * Per ordinare il preventivo, e' utile avere un vettore contenente i
         * costi dei singoli libri.
         *
         * Notare che questo vettore e' "parallelo" ai vettori del preventivo,
         * quindi nell'ordinare dovro' fare 3 scambi.
         */
        double[] prev_costi = new double[preventivo.length];

        /* calcolo i vari costi (simile alla funzione costo) */
        for (i = 0; i < preventivo.length; i++) {

            String titolo = preventivo[i].getNome();
            for (j = 0; j < libreria.length; j++) {
                if (titolo.equals(libreria[j].getNome())) {
                    // Nota: l'indice dei costi e' lo stesso del preventivo
                    prev_costi[i] = preventivo[i].getQuantita() *
libreria[j].getPrezzo();
                }
            }
        }

        /* ordino rispetto al nuovo vettore */
        for (i = 0; i < (prev_costi.length - 1); i++) {

            // cerco il minimo
            double minimo = prev_costi[i];
            int posizione = i;

            for (j = i + 1; j < prev_costi.length; j++) {

                if (minimo > prev_costi[j]) {
                    // Aggiorno il minimo
                    minimo = prev_costi[j];
                    posizione = j;
                }
            }

            // scambi
            double tmp_c = prev_costi[i];
            prev_costi[i] = prev_costi[posizione];
            prev_costi[posizione] = tmp_c;

            Prev_libro tmp_p = preventivo[i];
            preventivo[i] = preventivo[posizione];
            preventivo[posizione] = tmp_p;
        }
    }

    /** Calcola il costo di un libro */
    public static double costo_libro(Lib_libro[] libreria, Prev_libro libro) {
        double costo = 0;
        String titolo = libro.getNome();
        for (int j = 0; j < libreria.length; j++) {
            if (titolo.equals(libreria[j].getNome())) {
                costo = libro.getQuantita() * libreria[j].getPrezzo();
            }
        }
        return costo;
    }


    public static boolean[] validi(Lib_libro[] libreria, Prev_libro[] preventivo) {

        /*
         * Per risolvere questo quesito ho bisogno di conoscere il costo dei
         * singoli elementi e di sommare gli elementi a costo minore, finche' il
         * costo totale rimane <= 1000. Poiche' l'ordine e' crescente, gli
         * elementi a costo minore stanno nelle prime posizioni.
         */
```

```
		boolean[] prev_validi = new boolean[preventivo.length];
		double costo_totale = costo(libreria, preventivo);

		if (costo_totale <= 1000) {
			// tutti validi

			for (int i = 0; i < prev_validi.length; i++)
				prev_validi[i] = true;

		} else {

			/*
			 * Elimino gli elementi che mi rendono non valido il preventivo.
			 * Nota : questi elementi stanno alla fine.
			 */

			int i = preventivo.length - 1;
			while (costo_totale>1000) {
				// uso una funzione per calcolare il costo del libro
				costo_totale -= costo_libro(libreria, preventivo[i]);
				prev_validi[i] = false;
				i--;
			}

			// tutti gli altri elementi saranno validi
			for (; i >=0 ; i--) {
				prev_validi[i] = true;
			}

		}

		return prev_validi;
	}

	public static void main(String[] args) {
		Lib_libro[] libreria = new Lib_libro[] {
				new Lib_libro("Analisi I", 100),
				new Lib_libro("C++", 225),
				new Lib_libro("Visual Basic", 150),
				new Lib_libro("Visio", 300)};

		Prev_libro[] preventivo = new Prev_libro[3];
		preventivo[0] = new Prev_libro("C++", 5);
		preventivo[1] = new Prev_libro("Visual Basic", 1);
		preventivo[2] = new Prev_libro("Analisi I", 6);

		// costo
		double costo = costo(libreria, preventivo);
		System.out.println("Il costo totale e' " + costo);

		// ordinamento
		ordina(libreria, preventivo);
		System.out.println("Ordinato ");
		for (int i=0; i<preventivo.length; i++) {
			System.out.println("  " + preventivo[i].getNome() +
"\t"+preventivo[i].getQuantita());
		}

		// validita'
		boolean[] valid_1 = validi(libreria, preventivo);
		System.out.print("Validita' (1) =");
		for (int i=0; i<valid_1.length; i++) {
			System.out.print("  " + valid_1[i]);
		}
		System.out.println();
	}
}
```