

Studente (Cognome Nome): _____

Matricola: _____

Corso di Informatica
Corso di Laurea in Ingegneria Gestionale
a.a. 2007-08
Primo scritto – 11 Gennaio 2008

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

Quesito 1

I conti correnti presenti in una banca sono rappresentati da tre vettori contenenti rispettivamente il nome completo del titolare del conto, il codice numerico del conto e dal saldo.

a) Scrivere un metodo Java che cerchi il numero di conti correnti in rosso. Il metodo deve avere il seguente prototipo:

```
public static int conta_cc_rossi(String [] titolare, int [] codice,
int [] saldo)
```

Prerequisiti del metodo: gli array non sono null e hanno la stessa lunghezza.

b) Scrivere un metodo Java che esegua un prelievo su un conto se il saldo è superiore all'importo (solo in tal caso restituisce true). Il conto viene identificato da op_codice; se op_codice non è un codice valido (ossia non è presente nell'array dei codici di conto) il metodo non deve effettuare alcuna operazione e deve restituire false. Il metodo deve avere il seguente prototipo:

```
public static boolean prelievo(String [] titolare, int [] codice, int
[] saldo, int op_codice, int op_importo)
```

Prerequisiti del metodo: gli array non sono null e hanno la stessa lunghezza; op_importo è >0.

c) Scrivere un metodo Java che trova i conti correnti in rosso, restituendo i codici in un nuovo vettore ordinato a partire da quello maggiormente in rosso. Il metodo deve avere il seguente prototipo:

```
public static int [] prelievo(String [] titolare, int [] codice, int
[] saldo)
```

Prerequisiti del metodo: gli array non sono null e hanno la stessa lunghezza.

d) Scrivere un programma main di test che

Inizializza la banca con i seguenti valori:

Titolare	Codice	Saldo
Marco Rossi	1234	1400
Annalisa Bianchi	1599	-100
Marianna Rossi	109	-550
Andrea Ferrari	756	2400

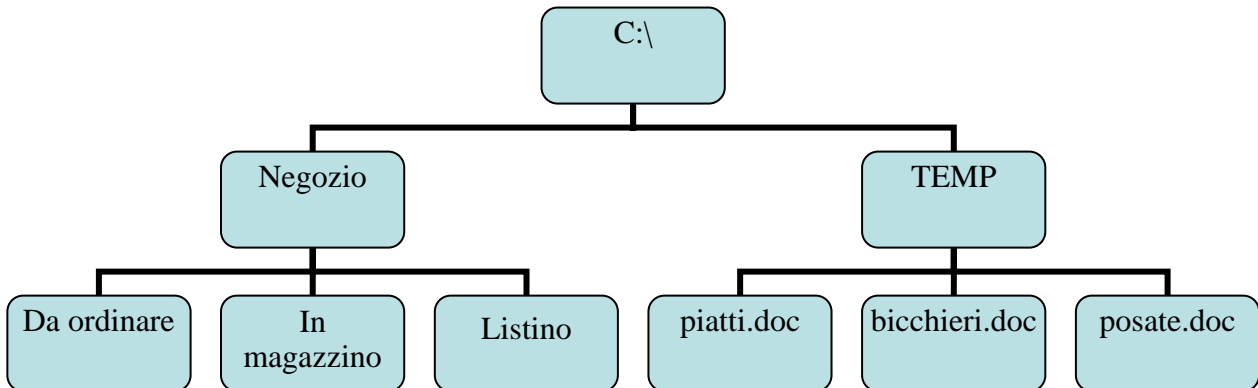
Stampare "conta rossi ok", se la funzione conta_cc_rossi (chiamata su questi vettori) restituisce 2. Altrimenti stampa "conta rossi fallito".

Chiama (per 2 volte) la funzione prelievo con i seguenti parametri, controllando che il valore restituito corrisponda a quello atteso, stampando "ok" o "errore"

op_codice	Importo	Valore atteso
1234	100	True
109	20	False

Quesito 2

Si consideri il seguente file system, di cui esistono solo la directory C:\ e TEMP, con i relativi file contenuti. Si tenga presente che “piatti” e “bicchieri” sono presenti in magazzino, mentre le “posate” sono da ordinare.



- Impartire i comandi per creare le directory mancanti, supponendo che la directory corrente sia C:\.
- Impartire la sequenza di comandi per spostare i file da C:\TEMP in “listino” e creare dei link all’interno delle 2 cartelle “da ordinare” e “in magazzino” verso i vari file, utilizzando solo path-name relativi. Si suppone che la directory corrente sia C:\TEMP. È possibile navigare fra le directory utilizzando il comando cd.

Quesito 3

- Rappresentare in C2 su 7 bit i seguenti numeri:

-26
70
-4

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

Quesito 1 – Risoluzione

```

public class PrimoScritto_genn {

    public static int conta_cc_rossi(String[] titolare, int[] codice, int[]
saldo) {
        int conti = 0;
        int i;
        for (i=0; i<saldo.length; i++) {
            if (saldo[i]<0)
                conti++;
        }
        return conti;
    }
}
  
```

```

    /*
    * Metodo di supporto: cerca un codice all'interno del vettore,
    restituendo l'indice
    * in cui e' stato trovato o -1 se il codice non e' presente
    */
    public static int cerca_codice(int[] codice, int op_codice) {
        // importante : inizializzo l'indice a -1 (utile se non trovo il
codice)
        int trovato = -1;
        int i;
        // continuo a cercare finche' non trovo il codice (trovato != -1)
        for (i=0; i<codice.length && trovato == -1; i++) {

            if (codice[i] == op_codice) {
                trovato = i;
            }

        }

        return trovato;
    }

    public static boolean prelievo_punto_B(String[] titolare, int[] codice,
int[] saldo,
        int op_codice, int op_importo) {

        boolean fatto = false;
        int posizione = cerca_codice(codice, op_codice);

        if (posizione != -1) {
            // esamino il saldo
            if (saldo[posizione] > op_importo) {
                saldo[posizione] -= op_importo;
                fatto = true;
            } // non e' necessario scrivere il ramo else,
            // perche' ho gia' inizializzato fatto a false

        } // non e' necessario scrivere il ramo else,
        // perche' ho gia' inizializzato fatto a false

        return fatto;
    }

    /*
    * Metodo di supporto che scambia 2 elementi all'interno dello stesso
vettore
    */
    public static void scambia(int[] vett, int pos_1, int pos_2) {

        int temp = vett[pos_1];
        vett[pos_1] = vett[pos_2];
        vett[pos_2] = temp;
    }

    public static int[] prelievo_punto_C(String[] titolare, int[] codice,
int[] saldo) {

        // prima di iniziare, serve sapere quanti sono i codici in rosso
        int num_cc_rossi = conta_cc_rossi(titolare, codice, saldo);

        // nota : creo 2 vettori (paralleli) per mantenere sia
l'informazione del codice
        // che del saldo di ciascun conto in rosso. In questo modo si
semplifica l'operazione
        // di ordinamento

```

```

int[] conti_rossi_codice = new int[num_cc_rossi];
int[] conti_rossi_saldo = new int[num_cc_rossi];

int i=0; // scorre i vettori con tutti i conti (rossi e non)
int j=0; // scorre i vettori conti_rossi...
// riempio i due vettori, scorrendo i vari conti alla ricerca di
quelli rossi
for (i=0; i<saldo.length; i++) {
    if (saldo[i]<0) {
        // trovato un nuovo conto in rosso
        conti_rossi_codice[j] = codice[i];
        conti_rossi_saldo[j] = saldo[i];
        j++;
    }
}

// ordino i vettori dei conti in rosso
for (i=0; i<conti_rossi_saldo.length; i++) {
    int pos_min = i;
    int min = conti_rossi_saldo[pos_min];

    for (j=i+1; j<conti_rossi_saldo.length; j++) {
        if (conti_rossi_saldo[j]<min) {
            // ho trovato un nuovo minimo
            pos_min = j;
            min = conti_rossi_saldo[pos_min];
        }
    }

    // scambio sia gli elementi presenti nel vettore dei codici
    // che quelli presenti nel vettore dei saldi, per mantenere
    // i 2 vettori paralleli
    scambia(conti_rossi_codice, i, pos_min);
    scambia(conti_rossi_saldo, i, pos_min);
}

return conti_rossi_codice;
}

/**
 * @param args
 */
public static void main(String[] args) {
    String[] titolari = new String[] {
        "Marco Rossi",
        "Annalisa Bianchi",
        "Marianna Rossi",
        "Andrea Ferrai"};

    int[] codici = new int[] {1234, 1599, 109, 756};
    int[] saldi = new int[] {1400, -100, -550, 2400};

    if (conta_cc_rossi(titolari, codici, saldi) == 2) {
        System.out.println("Conta rossi ok");
    } else {
        System.out.println("Conta rossi fallito");
    }

    if (prelievo_punto_B(titolari, codici, saldi, 1234, 100) == true) {
        System.out.println("ok");
    } else {
        System.out.println("errore");
    }
}

```

```

        if (prelievo_punto_B(titolari, codici, saldi, 109, 20) == false) {
            System.out.println("ok");
        } else {
            System.out.println("errore");
        }
    }
}

```

Quesito 2 – Risoluzione

a)

```

> mkdir Negozio
> cd Negozio
> mkdir "Da ordinare" "In magazzino" "Listino"

```

b)

```

> cd ..
> move TEMP\piatti.doc Negozio\Listino
> move TEMP\bicchieri.doc Negozio\Listino
> move TEMP\posate.doc Negozio\Listino
> cd Negozio\Listino
> linkd piatti.doc "..\In magazzino"
> linkd bicchieri.doc "..\In magazzino"
> linkd posate.doc "..\Da ordinare"

```

Quesito 3 - Risoluzione

a) L'intervallo di rappresentazione per i numeri in C2 su 7 bit è [-64, 63], per cui solo -26 e -4 sono rappresentabili.

Rappresentazione di -26: si parte dalla rappresentazione in notazione posizionale di 26 su 7 bit, utilizzando il metodo delle divisioni successive (che si omette per semplicità, ma nella soluzione va riportato), quindi si complementa:

$$26 = 0011010$$

$$-26 = \overline{0011010} + 1 = 1100101 + 1 = 1100110$$

Rappresentazione di -4: si parte dalla rappresentazione in notazione posizionale di 4 su 7 bit, utilizzando il metodo delle divisioni successive (che si omette per semplicità, ma nella soluzione va riportato), quindi si complementa:

$$4 = 0000100$$

$$-4 = \overline{0000100} + 1 = 1111011 + 1 = 1111100$$

b) (versione su 8 bit) su 8 bit tutti i numeri sono rappresentabili, per cui occorre rappresentare anche 70 in C2 su 8 bit: $70 = 01000110$. -26 e -4 vanno estesi su 8 bit. L'estensione avviene replicando il

segno: $-26=11100110$, $-4 = 11111100$.

allora le somme $70 + (-26)$ e $70 + (-4)$ sono sicuramente rappresentabili su b bit, perché i numeri sono discordi. Per la somma $-26 + (-4)$ occorre vedere il segno del risultato.

Calcolo delle somme:

$$70 + (-26) = 01000110 + 11100110 = 00101100$$

$$70 + (-4) = 01000110 + 11111100 = 01000010$$

$$(-26) + (-4) = 11100110 + 11111100 = 11100010$$

Poiché il risultato di $-26+(-24)$ è concorde con i due operandi la somma è rappresentabile.

A riprova, $00101100 = 44$, 01000010 vale 66 , e 1110010 vale -30

b) (versione su 7 bit, indicata come via alternativa in aula) su 7 bit l'unica operazione possibile è data da $-26+(-4)$. Poiché gli operandi sono concordi, occorre vedere il segno del risultato.

$$(-26) + (-4) = 1100110 + 1111100 = 1100010$$

Poiché il risultato di $-26+(-24)$ è concorde con i due operandi la somma è rappresentabile.

A riprova, 110010 vale -30 in C2 su 7 bit