Fondamenti di Informatica L-A (A.A. 2004/2005) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – I Prova Intermedia del 11/11/2004 - durata 2h - COMPITO A

- **Esercizio 1 (punti 12)**

Un commerciante vuole tenere sotto controllo i prezzi dei suoi prodotti. A tal scopo vuole realizzare un programma che chieda, per un numero finito di prodotti, il nome del prodotto e il suo prezzo in euro, e restituisca come output il nome del prodotto più costoso e il prezzo incriminato.

A tal scopo si definisca inizialmente una struttura di nome **`prodotto`**, composta dal nome del prodotto (al più 127 caratteri) e dal prezzo (rappresentato tramite **`float`**). Si realizzi poi:

a) una procedura
```
void calcola(prodotto lista[], int length, char piucaro[], float *prezzo)
```
che riceve in ingresso un array **`lista`** di tipo **`prodotto`**, e la lunghezza **`length`** di tale array. La funzione deve determinare il prodotto più costoso e copiarne il nome nella stringa **`piucaro`**; il costo in euro deve poi essere restituito tramite la variabile **`prezzo`** (passata per riferimento). Al fine di copiare il nome del prodotto più costoso, si utilizzi la funzione di libreria **`strcpy(char *s, char *ct)`**, che copia la stringa **`ct`** nella stringa **`s`**, terminatore compreso. **(punti 6)**

b) un programma **`main()`** che chieda all'utente di inserire, alternativamente, il nome di un prodotto e il suo prezzo, per un totale di **`DIM`** prodotti (si controlli che il prezzo sia maggiore o uguale a 0; in caso contrario si ripeta la richiesta di inserire il prezzo). Il programma deve provvedere a memorizzare tali informazioni in un array di tipo **`prodotto`** precedentemente definito, di dimensione fissa **`DIM`** (si supponga per semplicità che **`DIM`** valga 10). Una volta raccolte le informazioni, si utilizzi la procedura **`calcola()`** definita al punto precedente per ottenere il nome e il prezzo del prodotto più costoso, e si stampino a video tali informazioni. **(punti 6)**

- **Esercizio 2 (punti 7)**

Dato il seguente programma C:

```
#include <stdio.h>

int rotate(int * nome, int length);

int main() {
	int temp[] = {1, 2, 3, 4, 5, 6};
	int i= 0, result;

	result = rotate(temp, 3);

	for (i=0; i<result; i++)
		printf("%d\n", temp[i]);

	return 0;
}

int rotate(int * nome, int length) {
	int i = 0;
	char temp;

	temp = nome[0];
	while (i<length-1) {
		*nome = *(nome+1);
		++i;
		nome++;
	}
	*nome = temp;
	return length;
}
```

Che cosa fa la funzione **`rotate()`**? Che cosa stampa il programma? Si motivi opportunamente la risposta.

- **Esercizio 3 (punti 5)**

Si consideri la seguente funzione:

```
int G(int a, int b) {
       if (a < b )
              return b*a;
       else {
              b++;
              a--;
              return G(a, b);
       }
}
```

Si scriva il risultato della funzione quando invocata come `G(4, 0)` e si mostrino i record di attivazione. La funzione è tail-ricorsiva?

- **Esercizio 4 (punti 6)**

Si scriva una funzione ricorsiva `int somma(int x)` che, ricevuto come parametro un numero intero positivo (pari o dispari), calcoli ricorsivamente la somma di tutti i numeri pari compresi tra 0 e quel numero (compreso).

- **Esercizio 5 (punti 2)**

Un elaboratore rappresenta i numeri interi su 8 bit dei quali 7 sono dedicati alla rappresentazione del modulo del numero e uno al suo segno. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

Soluzione Compito A

Esercizio 1

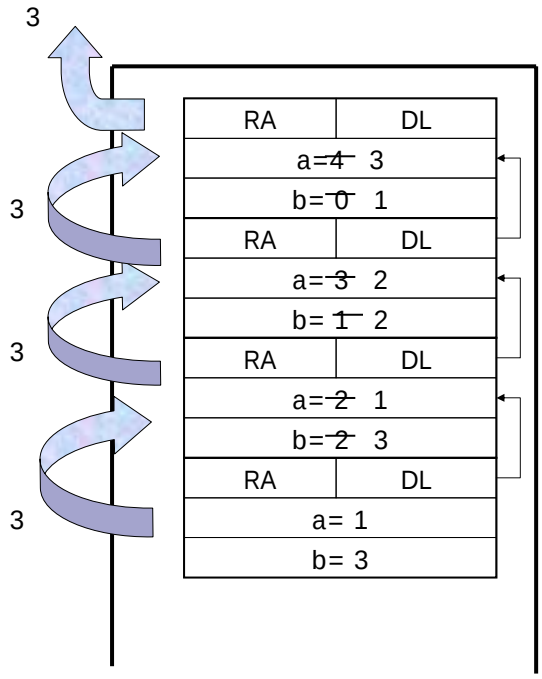
```
#include <stdio.h>
#include <string.h>

#define MAX 128
#define DIM 10

typedef struct {
	char nome[MAX];
	float prezzo;
} prodotto;

void calcola(prodotto lista[], int length, char piucaro[], float * prezzo) {
	int i;
	int pos = 0;

	if (length <= 0) {  // controllo non richiesto dal testo...
		strcpy(piucaro, "ERRORE");
		*prezzo = 0;
		return;
	}
	for (i=1; i<length; i++)
		if (lista[i].prezzo > lista[pos].prezzo) {
			pos = i;
		}
	strcpy(piucaro, lista[pos].nome);
	*prezzo = lista[pos].prezzo;
}

int main() {
	int i;
	prodotto lista[DIM];
	char piucaro[MAX];
	float prezzo = 0;

	for (i=0; i<DIM; i++) {
		printf("Inserire nome prodotto: ");
		scanf("%s", lista[i].nome);
		do {
			printf("Inserire prezzo: ");
			scanf("%f", &lista[i].prezzo);
		} while (lista[i].prezzo < 0);
	}

	calcola(lista, i, piucaro, &prezzo);
	printf("Il prodotto piu' caro e' il %s, che costa %6.2f euro.\n", piucaro,
prezzo);
	return 0;
}
```

Esercizio 2

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
2 3 1
All'inizio del **`main`** sono dichiarate tre variabili, **`temp`** come array di interi, **`result`** e **`i`** come interi. Subito dopo questa fase a **`result`** è assegnato il valore di ritorno della funzione **`rotate`**. Viene eseguito un ciclo con **`i`** come indice che va da 0 fino al valore indicato da **`result`** (escluso); questo ciclo si limita a stampare a video i primi interi contenuti in **`temp`**, e poi termina. Dunque, al fine di comprendere che cosa stampi il programma, è necessario comprendere quali operazioni esegue la funzione **`rotate()`**, e quale valore ritorna.

La funzione **`rotate()`** riceve in ingresso un puntatore ad intero chiamato **`nome`**, ed un intero **`length`**. Come prima operazione, dopo aver dichiarato alcune variabili, assegna a **`temp`** (definito locale) il valore intero puntato da **`nome`**. Quindi esegue un ciclo, in cui ad ogni passo assegna alla cella di memoria puntata da **`nome`** il valore della cella di memoria puntata da **`(nome+1)`**. Poiché **`nome`** è dichiarato come puntatore ad intero, il compilatore interpreta l'operazione di somma come incremento del puntatore affinché esso punti il prossimo intero in memoria. Avendo passato come argomento un array di interi, ciò significa che alla posizione 0 dell'array viene assegnato il valore presente nella posizione 1; alla posizione 1 dell'array viene assegnato il valore presente nella posizione 2, e così via ad ogni ciclo dell'istruzione **`while`**. Effettuato l'assegnamento, le variabili **`i`** e **`nome`** vengono incrementate (rispettivamente pre- e post- incrementate), e poi si riesegue il ciclo fino a che la condizione del **`while`** diventa falsa. Poiché **`rotate()`** viene invocata con **`length`** pari a 3, il ciclo viene eseguito 2 volte. Come ultima operazione all'ultima posizione puntata da **`nome`** viene assegnato il valore salvato all'inizio nella variabile di appoggio **`temp`**; viene poi restituito il valore **`length`**.
La funzione **`rotate()`** quindi realizza la rotazione dei primi elementi di un array passato come parametro; il numero di elementi da ruotare è definito tramite la variabile **`length`**, anch'essa passata come parametro alla funzione. In questo caso specifico vengono ruotati i primi tre valori dell'array **`temp`** definito nel main, ottenendo quindi 2, 3, 1.
La funzione **`main()`** si limita a stampare i "**`result`**" elementi, dove **`result`** viene a valere 3. Quindi stampa, su tre righe distinte, i valori 2, 3, 1.

Esercizio 3
Il risultato è 3.



La funzione è tail-ricorsiva.

Esercizio 4

```
int somma(int x) {
	if (x == 0)
		return 0;
	else {
		if (x%2 == 0) return x + somma(x-2);
		else return somma(x-1);
	}
}
```

Esercizio 5

39->	0	0100111
109->	0	1101101
Tra i moduli dei numeri si esegue una sottrazione ottenendo:	1	1000110
che vale -70 in base dieci.