Fondamenti di Informatica L-A (A.A. 2005/2006) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame del 29/03/2006 - durata 2h30m
COMPITO A

ESERCIZIO 1 (13 punti)
Una piccola ditta produce e vende piastrelle quadrate per pavimenti. Per ogni lotto di piastrelle presente in magazzino, la ditta ha salvato, su un file binario di nome "**`products.bin`**", strutture dati denominate **`tilesBatch`**, contenenti: un codice identificativo del lotto (una stringa ben formata di 6 caratteri), un codice identificativo del tipo di piastrella (una stringa ben formata di 6 caratteri), la dimensione del lato della piastrella (un intero, inteso come lunghezza in cm), e la quantità di piastrelle disponibili in tale lotto (ancora un intero). Non è noto a priori quanti lotti siano registrati nel file, ma è certo che nel file ci sono più lotti relativi allo stesso tipo di piastrella. La ditta vuole sviluppare un programma per la vendita al cliente: il cliente deve inserire il codice della piastrella scelta, e le dimensioni (in cm) del pavimento. Il programma deve calcolare una stima di quante piastrelle sono necessarie, e stabilire se ve ne sono abbastanza in magazzino. Dopo aver definito opportunamente la struttura dati **`tilesBatch`**, il candidato realizzi:

1. una funzione
   `tilesBatch * extractStock(FILE * production, char * tileCode, int* length)`

   che, ricevuto in ingresso un puntatore a file, e il codice della piastrella scelta, restituisca in un'area di memoria opportunamente allocata le strutture **`tilesBatch`** che si riferiscono alla piastrella scelta; tramite il parametro **`length`**, la funzione deve restituire il numero di strutture dati **`tilesBatch`** allocate in memoria; (5 punti)
2. un programma **`main()`** che apra in modo opportuno il file "**`products.bin`**", e quindi richieda all'utente il codice della piastrella scelta e le dimensioni (in cm, valori **`float`**) del pavimento da piastrellare. Il programma utilizzi la funzione **`extractStock(…)`** per estrarre dal file i lotti corrispondenti al tipo di piastrella scelto, e poi calcoli (e stampi a video) quante piastrelle sono necessarie al cliente. Si determini poi se vi sono in magazzino sufficienti piastrelle per esaudire la richiesta del cliente; in caso affermativo, il programma deve stampare anche i codici dei lotti necessari, selezionando i lotti in base ad un ordine qualsiasi (si consiglia, per semplicità, di selezionare i lotti secondo l'ordine con cui sono stati estratti dalla funzione di cui al punto 1, cioè di scegliere i primi n lotti che forniscono un numero totale di piastrelle sufficiente). (8 punti)

Si ricorda l'esistenza della funzione **`void rewind(*FILE)`** che riporta la testina di lettura ad inizio file, e della funzione **`int strcmp(char* st, char * ct)`** per il confronto tra stringhe. Al fine di determinare quante piastrelle siano necessarie, si adotti il seguente criterio: se un lato è lungo **`x`** cm e una piastrella è lunga **`y`** cm, allora lungo quel lato serviranno almeno **`x/y+1`** piastrelle. Il numero di piastrelle totali è dato dal prodotto delle piastrelle necessarie per ogni lato, più un 10% del totale (come margine di sicurezza).

ESERCIZIO 2 (9 punti)
Si scriva una funzione **`int isDiff(list l1, list l2, list diff)`** che, ricevute in ingresso tre liste di interi, senza ripetizioni ed ordinate, determini se la lista **`diff`** è la differenza tra la lista **`l1`** e la lista **`l2`**. In particolare la funzione deve restituire un valore maggiore di 0 se effettivamente facendo il "merge" delle liste **`l2`** e **`diff`** si ottiene nuovamente la lista **`l1`**; la funzione deve ritornare 0 altrimenti.
Ad esempio, date le liste **`l1=[1,2,3,4]`**, **`l2=[1,4]`**, **`diff=[2,3]`**, la funzione deve restituire valore 1. Invece, se invocata con **`l1=[1,2,3,4]`**, **`l2=[1,4]`**, **`diff=[3]`**, allora la funzione deve restituire 0.
La funzione deve essere realizzata utilizzando il tipo di dato astratto **`list`**, definito per gli interi (non è necessario riportare la definizione nella soluzione). Si possono utilizzare le sole operazioni primitive definite durante il corso, che quindi possono NON essere riportate nella soluzione. Non si possono usare altre funzioni di alto livello.

ESERCIZIO 3 (6 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
char * temp  = "Tr";

int lookFor(char* el, char* str){
      char * temp;
      int inSequence=0, found=0;
      while (*str!='\0') {
            temp = el;
            if (*str == temp[0]) {
                  inSequence = 1;
                  while ((*str!='\0') && (temp[0]!='\0') && inSequence) {
                        if (*str == temp[0])    {     str++; temp++; }
                        else inSequence--;
                  }
                  if (inSequence && temp[0]=='\0') {  found++; }
            }
            else str = str + 1;
      }
      return found;
}

int main() {
      char * sentence = "Trentatre' Trentini", * el;
      int i;
      el = malloc(sizeof(char) * 3);
      i = lookFor(temp,sentence);
      printf("%s %d\n",temp,i);
      free(el);
      return 0;
}
```

ESERCIZIO 4 (4 punti)

Data la funzione:

```
int duplica(int a, int b) {
      int temp;

      if (a>=0)
            a--;
      if (a/b == 0) return 1;
      else {
            b++;
            return b+duplica(a-b, b);
      }
}
```

e la funzione chiamante:

```
int main()
{
  printf("%d\n", duplica(9,2));
  return 0;
}
```

mostrare la sequenza dei record di attivazione. Che cosa viene stampato sullo standard output?

SOLUZIONE

Esercizio1

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

typedef struct {
	char batchCode[7];
	char tileCode[7];
	int sideLength;
	int quantity;
} tilesBatch;

tilesBatch * extractStock(FILE * production, char * tileCode, int* length) {
	tilesBatch * batches, temp;
	int max=0;

	while (fread(&temp, sizeof(tilesBatch), 1, production)>0)  max++;
	rewind(production);

	batches = (tilesBatch *) malloc( sizeof(tilesBatch) * max);
	*length = 0;
	while(fread((batches+(*length)), sizeof(tilesBatch), 1, production)>0)
		if (strcmp(tileCode, batches[*length].tileCode)==0)
			*length = *length + 1;
	return batches;
}

int main(){
	FILE* fPtr;
	tilesBatch * stock;
	char tilesCode[7];
	float side_a, side_b;
	int size_a, size_b, length;
	int tilesNeeded, i, tot=0;

	if((fPtr=fopen("production.bin","rb"))==NULL){
			printf("Problemi durante l'apertura del file products.bin\n");
			exit(-1); }
	printf("Inserire il codice della piastrella: ");
	scanf("%s", tilesCode);
	stock = extractStock(fPtr, tilesCode, &length);

	if (length > 0) {
		printf("Inserire le dimensioni del pavimento: ");
		scanf("%f%f", &side_a, &side_b);

		size_a = ((int) side_a / stock[0].sideLength);
		size_b = ((int) side_b / stock[0].sideLength);
		tilesNeeded = ((size_a+1)*(size_b+1)*110)/100;
		printf("Piastrelle necessarie: %d\n", tilesNeeded);

		for (i=0; i<length && tot<tilesNeeded; i++)
			tot = tot + stock[i].quantity;
		if (tot < tilesNeeded)
			printf("Non ci sono abbastanza piastrelle\n");
		else {
			printf("Sono necessari i seguenti lotti di piastrelle:\n");
			for (i=0; i<length && tilesNeeded>0; i++) {
				printf("Lotto cod.: %s\n", stock[i].batchCode);
				tilesNeeded = tilesNeeded - stock[i].quantity;
			}
		}
	}
	else printf("Non ci sono piastrelle del tipo indicato.\n");
	fclose(fPtr);
	return 0; }
```

Esercizio2

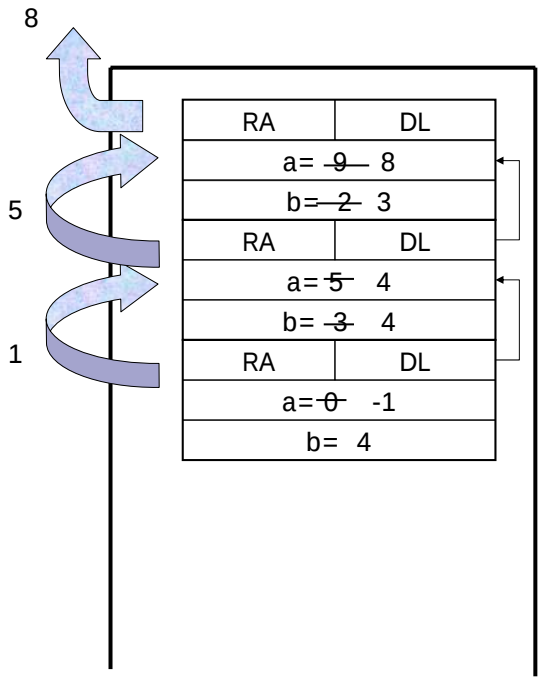
```
int isDiff(list l1, list l2, list diff) {
	int problem = 0;

	while (!empty(l1) && !problem) {
		if ( (!empty(l2)) && (head(l1) == head(l2)) ) {
			l1 = tail(l1);
			l2 = tail(l2);
		}
		else if ((!empty(diff)) && (head(l1) == head(diff)) ) {
				l1 = tail(l1);
				diff = tail(diff);
			}
			else problem=1;
	}
	if (!problem && empty(l1) && empty(l2) && empty(diff)) return 1;
	else return 0;
}
```

Esercizio3

La funzione **`lookFor(...)`** prende in ingresso due stringhe ben formate, e conta quante volte la prima stringa compare nella seconda. Viene invocata con parametri **`temp`** (globale, che vale "**`Tr`**") e **`sentence`** (che vale "**`Trentatre' Trentini`**"). "Tr" compare due volte, e quindi la funzione restituisce il valore 2.
Il main alloca dinamicamente spazio per 3, ma poi non utilizza tale spazio, e alla fine si limita a deallocarlo.Invoca la funzione **`lookFor`**, e stampa a video la sottostringa cercata, ed il numero di volte in cui l'ha trovata. Sullo standard output quindi viene stampato:

`Tr 2`

Esercizio 4



Sullo standard output viene scritto il risultato della chiamata alla funzione, ovvero:

8