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 B

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 "**`produzione.dat`**", strutture dati denominate **`prodStock`**, contenenti: un codice identificativo del lotto (una stringa ben formata di 4 caratteri), un codice identificativo del tipo di piastrella (una stringa ben formata di 4 caratteri), la dimensione del lato della piastrella (un float, inteso come lunghezza in cm), e la quantità di piastrelle disponibili in tale lotto (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 avere definito opportunamente la struttura dati **`prodStock`**, il candidato realizzi:

1. una funzione

   ```
   prodStock * findTiles(char * fileName, char * tileCode, int* length)
   ```

   che, ricevuto in ingresso il nome di un file ed il codice della piastrella scelta, apra il file, ne acceda il contenuto, e restituisca in un'area di memoria opportunamente allocata le strutture **`prodStock`** che si riferiscono alla piastrella scelta; tramite il parametro **`length`**, la funzione deve restituire il numero di lotti effettivamente allocati in memoria; (6 punti)
2. un programma **`main()`** che richieda all'u-tente il codice della piastrella scelta e le dimensioni (in cm, un intero) del pavimento da piastrellare. Il programma utilizzi la funzione **`findTiles(…)`** per estrarre dal file "**`produzione.dat`**" i lotti corrispondenti al tipo di piastrella scelto, e poi calcoli (e stampi a video) quante piastrelle sono necessarie al cliente. Si determini poi quanti lotti vi sono in magazzino, tali che, presi singolarmente, abbiano un numero di piastrelle sufficienti per coprire il pavimento senza ricorrere ad altri lotti. Qualora esistano tali lotti, il programma stampi a video anche i codici identificativi di tali lotti. (7 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: si calcoli l'area del pavimento da coprire, e si divida tale area per l'area di una singola piastrella. Si arrotondi poi all'intero superiore il risultato (sommandovi 1): il numero di piastrelle necessarie è dato da tale valore, aumentato del 20% come margine di sicurezza.

ESERCIZIO 2 (9 punti)
Si scriva una funzione **`list diff3(list l1, list l2, list l3)`** che, ricevute in ingresso tre liste di interi, senza ripetizioni ed ordinate, restituisca una nuova lista contenente gli elementi di **`l1`** che non sono presenti nemmeno una volta in **`l2`** o in **`l3`**. Qualora gli elementi di **`l1`** siano tutti presenti in **`l2`** e **`l3`**, la funzione deve restituire la lista vuota.
Ad esempio, date le liste **`l1=[1,2,3,4]`**, **`l2=[1,4]`**, **`l3=[2,3]`**, la funzione deve restituire **`[]`**. Invece, se invocata con **`l1=[1,2,3,4]`**, **`l2=[1,4]`**, **`l3=[3]`**, allora la funzione deve restituire **`[2]`**.
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  = "os";

int lookFor(char* el, char* str){
      char * temp;
      int found=0, count=0;

      while (*str!='\0') {
            temp = el; found=0;
            while((temp[0] != '\0') && !found) {
                  if (*str == temp[0]) {
                        count++;
                        found = 1;
                  }
                  temp = temp+1;
            }
            str++;
      }
      return count; }

int main() {
      char * sentence = "Rosso di sera";
      char * 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:

```
float dup(float num1, float num2) {

      num1 = num1 + 2;
      if (num1>=num2 ) return 1.0;
      else {
            num2++;
            return num2+dup(num1, num2);
      }
}
```

e la funzione chiamante:

```
int main()
{
  printf("%f\n", dup(1,6));
  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[5];
	char tileCode[5];
	float sideLength;
	int quantity;
} prodStock;

prodStock * findTiles(char * fileName, char * tileCode, int* length) {
	FILE * fPtr;
	prodStock * batches;
	prodStock temp;
	int max=0;

	if((fPtr=fopen(fileName,"rb"))==NULL){
			printf("Problemi durante l'apertura del file %s\n", fileName);
			exit(-1); }

	while (fread(&temp, sizeof(prodStock), 1, fPtr)>0)  max++;
	rewind(fPtr);
	batches = (prodStock *) malloc( sizeof(prodStock) * max);

	*length = 0;
	while(fread((batches+(*length)), sizeof(prodStock), 1, fPtr)>0)
		if (strcmp(tileCode, batches[*length].tileCode)==0)
			*length = *length + 1;
	fclose(fPtr);
	return batches;
}

int main(){
	prodStock * stock;
	char tilesCode[5];
	int side_a, side_b, area, minTiles, length;
	float areaTile;
	int tilesNeeded, i, disp=0;

	printf("Inserire il codice della piastrella: ");
	scanf("%s", tilesCode);
	stock = findTiles("produzione.dat", tilesCode, &length);

	if (length > 0) {
		printf("Inserire le dimensioni del pavimento: ");
		scanf("%d%d", &side_a, &side_b);
		area = side_a*side_b;
		areaTile = stock[0].sideLength * stock[0].sideLength;
		minTiles = ((int) area/areaTile) + 1;
		tilesNeeded = (minTiles*120)/100;
		printf("Piastrelle necessarie: %d\n", tilesNeeded);

		for (i=0; i<length; i++)
			if (stock[i].quantity >= tilesNeeded) disp = disp + 1;
		if (disp == 0)
			printf("Non ci sono lotti che contengano sufficienti piastrelle\n");
		else {
			printf("Si possono usare i seguenti lotti di piastrelle:\n");
			for (i=0; i<length && tilesNeeded>0; i++)
				if (stock[i].quantity >= tilesNeeded)
					printf("Lotto cod.: %s\n", stock[i].batchCode);  }
	}
	else printf("Non ci sono piastrelle del tipo indicato.\n");
	return 0; }
```

Esercizio2

```
list diff3(list l1, list l2, list l3) {
      list result = emptylist();
      while (!empty(l1)) {
            if ( (!empty(l2)) &&  (head(l1) == head(l2)) ) {
                  l1 = tail(l1);
                  l2 = tail(l2);
            }
            else if ((!empty(l3)) && (head(l1) == head(l3)) ) {
                                    l1 = tail(l1);
                                    l3 = tail(l3);
                        }
            else {
                  result = cons(head(l1), result);
                  l1 = tail(l1);
            }
      }
      return result;
}
```

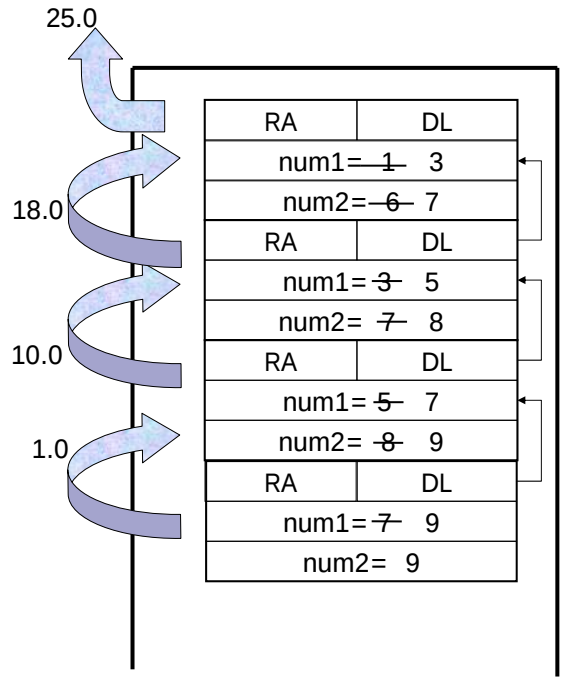
Esercizio3

La funzione **`lookFor(...)`** prende in ingresso due stringhe ben formate, e conta quante volte i caratteri della prima stringa compaiono nella seconda, eventualmente non in ordine. Viene invocata con parametri **`temp`** (globale, che vale "**`os`**") e **`sentence`** (che vale "**`Rosso di sera`**"). 'o' compare due volte, ed 's' compare tre volte: la funzione restituisce quindi il valore 5.

Il **`main()`** alloca dinamicamente spazio per 3 caratteri, ma poi non utilizza tale spazio, e alla fine si limita a deallocarlo. Invoca la funzione **`lookFor()`** e stampa a video l'insieme di caratteri cercati e il numero di volte che sono stati trovati. Sullo standard output quindi viene stampato:

```
os 5
```

Esercizio 4



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

```
25.0
```