Fondamenti di Informatica L-A (A.A. 2005/2006) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame del 11/07/2006 - durata 2h30m

ESERCIZIO 1 (13 punti)

Lo stabilimento balneare Granchio usa fare credito ai propri clienti dei prodotti in vendita. Il bagno vende 20 prodotti, etichettati con codici da 0 a 19, e tiene traccia dei crediti tramite apposite strutture dati di nome **`credito`** salvate su un file binario di nome "**`credits.dat`**". La struttura dati contiene il nome di un cliente (al più 15 caratteri) e un **`array`** di interi, di dimensione 20, dove in ogni posizione è segnata la quantità di prodotti consumati relativamente al prodotto identificato dall'indice dell'array. Ad esempio, in prima posizione, cioè con indice 0, è sempre segnata la quantità di prodotti di tipo 0 che un certo cliente ha consumato. Il candidato deve realizzare un programma che, chiesti in ingresso una lista di prodotti di cui si vuole recuperare il credito, stampi a video il nome dei clienti e la cifra dovuta da ognuno. A tal scopo, dopo aver definito opportunamente la struttura dati **`credito`**, il candidato realizzi:

1. una funzione
   `credito * getPeople(FILE * dati, int itemId[], int numItem, int * maxClienti)`

   che, ricevuto in ingresso un puntatore a file, un array di codici di prodotti **`itemId`** e la sua dimensione **`numItem`**, restituisca in un'area di memoria dinamicamente allocata le strutture **`credito`** relative ai clienti che hanno consumato almeno uno dei prodotti indicati (cioè la cui quantità consumata è maggiore di 0). Tramite **`maxClienti`**, la funzione deve restituire il numero di clienti creditori identificati; (6 punti)
2. un programma **`main()`** che apra in modo opportuno il file "**`credits.dat`**" e poi chieda all'utente quanti prodotti vuole verificare. Il programma allochi dinamicamente memoria sufficiente per memorizzare i codici identificativi dei prodotti, i loro prezzi (di tipo **`float`**), e poi chieda all'utente di specificare in ordine il codice di ogni prodotto seguito dal suo prezzo. Si invochi poi la funzione **`getPeople(…)`**, al fine di ottenere la lista dei creditori, e poi si calcoli (e si stampi a video) per ogni creditore la somma dovuta, utilizzando i prezzi precedentemente richiesti. (7 punti)

Si ricorda l'esistenza della funzione **`void rewind(*FILE)`** che riporta la testina di lettura ad inizio file.

ESERCIZIO 2 (8 punti)

Si scriva una funzione ricorsiva **`list clean(list)`** che, ricevuta in ingresso una lista di stringhe ben formate, restituisca una lista contenente le stesse stringhe senza ripetizioni. Ad esempio, data la lista **`l1= [“Federico”, “Marco”, “Fabrizio”, “Federico”]`**, la lista risultante deve essere **`[ “Marco”, “Fabrizio”, “Federico”]`** (l'ordine degli elementi della lista risultante non è rilevante). A tal scopo, si definisca anche una opportuna funzione iterativa **`int member(element el, list l1)`** che restituisce **`true`** se l'elemento **`el`** è contenuto in **`l1`**. Si tenga conto dei seguenti vincoli implementativi:

1. la funzione **`clean(…)`** deve essere realizzata ricorsivamente, facendo utilizzo delle sole operazioni primitive viste durante il corso e della suddetta funzione **`member(…)`**; non si possono usare altre funzioni di alto livello, salvo che il candidato le definisca nell'elaborato. (punti 5)
2. la funzione **`member(…)`** deve essere realizzata iterativamente, accedendo alle liste tramite la notazione a puntatori. (punti 3)

Le funzioni devono essere realizzate utilizzando il tipo di dato astratto **`list`**, definito per le stringhe (non è necessario riportare la definizione nella soluzione). Il candidato assuma che il tipo **`element`** sia definito come puntatore a carattere, e, se necessario, utilizzi la funzione di libreria **`strcmp(…)`** al fine di determinare se due stringhe sono identiche.

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 = 'e';
char word[] = "matterhorn";

int sub(char el, char* word){
      char temp;
      int i=0, j=6;

      for (i=0; *word != '\0'; i++) {
            for(j=0; j<i; j++) {
                  if (el == *(word-j)) {
                        temp = word[0];
                        *word = *(word-1);
                        *(word-1) = temp;
                  }
            }
            word++;
      }
      return i;
}

int main() {
      char word[] = {'e','i','g','e','r','\0'};
      int result;

      result = sub(*word,word);
      printf("%s %d\n",word, result);

      return 0;
}
```

ESERCIZIO 4 (2 punti)

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:

57-75

ESERCIZIO 5 (3 punti)

Si presenti brevemente come il linguaggio C effettua il passaggio di parametri nella invocazione di funzione/procedura. In particolare, si illustri che cosa succede nel caso di invocazione con passaggio del nome di un array e del nome di una struct.

SOLUZIONE

Esercizio1

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

typedef struct {
	char client[16];
	int quant[20];
} credito;

credito * getPeople(FILE * dati, int itemId[], int numItem, int * maxClienti) {
	int numClienti = 0, i, trovato;
	credito temp;
	credito * result;

	*maxClienti = 0;
	while (fread(&temp, sizeof(credito), 1, dati)>0) {
		trovato = 0;
		for (i=0; i<numItem && !trovato; i++)
			if (temp.quant[itemId[i]] > 0) {
				(*maxClienti)++;
				trovato = 1; }
	}
	rewind(dati);
	result = (credito *) malloc(sizeof(credito)* *maxClienti);
	while (fread((result+numClienti), sizeof(credito), 1, dati)>0) {
		trovato = 0;
		for (i=0; i<numItem && !trovato; i++)
			if ((result+numClienti)->quant[itemId[i]] > 0) {
				numClienti++;
				trovato = 1; }
	}
	return result; }


int main(){

	FILE * fPtr;
	int numItem, i, j, numPeople, * itemId;
	float * prices, money;
	credito * people;

	printf("Numero di beni da recuperare?");
	scanf("%d", &numItem);
	itemId = (int *)   malloc(sizeof(int)   * numItem);
	prices = (float *) malloc(sizeof(float) * numItem);

	printf("Inserire i codici dei beni e il prezzo di ognuno: ");
	for (i=0; i<numItem; i++) scanf("%d %f", itemId+i, prices+i);

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

	people = getPeople(fPtr, itemId, numItem, &numPeople);
	for (i=0; i<numPeople; i++) {
		money = 0;
		for (j=0; j<numItem; j++)
			money = money + people[i].quant[itemId[j]] * prices[j];
		printf("Sig. %s: %f euro\n", people[i].client, money); }

	fclose(fPtr);
	free(itemId); free(prices); free(people);
	return 0;
}
```

Esercizio2

```
int member(element el, list l1) {
	while (l1 != NULL) {
		if (strcmp(l1->value, el) == 0) return 1;
		else l1 = l1->next;
	}
	return 0; }

list clean(list l1) {
	list result = emptylist();
	char * temp;

	if (empty(l1)) return result;
	else {
		temp = head(l1);
		if (member(temp, tail(l1))) return clean(tail(l1));
		else return cons(temp, clean(tail(l1)) );
	}
}
```

Esercizio3

La funzione `sub(…)` prende in ingresso un carattere e una stringa, e poi effettua un ciclo su ogni carattere componente la stringa. Per ogni carattere, la funzione effettua un ulteriore ciclo alla ricerca dell'elemento passato come parametro; in tale ciclo, in ordine inverso, vengono controllati i caratteri precedenti al carattere attuale. Da notare che, siccome l'indice j è sempre strettamente minore dell'indice i, non viene mai controllato il primo carattere.
Quando il carattere passato come parametro (nel caso specifico, 'e') viene trovato nella stringa, il carattere corrente viene scambiato con il precedente. Per come sono stati vincolati gli indici, ciò avviene solo quando il carattere corrente è 'r' (cioè quando il puntatore a carattere word punta all'elemento 'r'). Quindi 'r' ed 'e' vengono scambiati, e la stringa viene modificata in "eigre". Viene restituito infine il valore dell'indice i, che vale 5.
Sullo standard output viene quindi stampato:
`eigre 5`

Esercizio 4

Sottrazione dei moduli (numero maggiore meno numero minore)

1001011-	75-
0111001=	57=
---	---
0010010	18

Aggiunta del segno
10010010