Fondamenti di Informatica L-A (A.A. 2006/2007) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Giovedì 12 Luglio 2007 – durata 2.30h

ESERCIZIO 1 (9 punti)

Un'associazione di volontari raccoglie firme per un referendum. Ogni firmatario viene registrato in una struttura dati `firma`, contenente due stringhe `nome` e `cognome`, ognuna al massimo di 31 caratteri (il candidato mostri nell'elaborato la definizione di tale struttura dati). Purtroppo capita che alcuni firmatari firmino per lo stesso referendum più volte e inoltre accade che alcuni volontari scambino inavvertitamente nomi con cognomi. Il candidato realizzi una funzione:

```
firma * eliminaDup(firma * l1, int dim1, firma * l2, int dim2, int * newDim)
```

che, ricevuti come parametri due vettori di strutture dati `firma` e le loro dimensioni, restituisca come risultato un nuovo vettore di strutture dati `firma` allocato dinamicamente e la sua dimensione logica `newDim`. Tale vettore deve contenere le strutture dati relative a tutte le firme contenute in `l1` e `l2` senza ripetizioni.

Si consiglia, per semplicità, di scrivere una funzione ausiliaria `confronta(…)` che, ricevute in ingresso due strutture dati di tipo `firma`, restituisce un valore vero se le firme sono della stessa persona (tenendo conto che ogni tanto nome e cognome sono scambiati), falso altrimenti.

VARIANTE SOLO PER VECCHIA MODALITÀ: Ogni banchetto registra in file di testo nome e cognome (o viceversa) di ogni firmatario (uno per ogni riga): in ogni file al massimo sono registrati 100 firmatari (si supponga che ogni nome e cognome non contengano internamente alcuno spazio). Il comitato organizzatore centrale poi possiede un file di testo "`files.txt`" dove in ogni riga sono registrati i nomi dei file dei vari banchetti. Si realizzi un programma che, letti dal file i nomi dei vari elenchi, provveda a leggere tutti i nomi in ogni file e, utilizzando la funzione `eliminaDup(…)`, calcoli un unico array di strutture dati `firma` contenente l'elenco di tutti i firmatari. Il contenuto di tale array deve infine essere stampato a video. Il candidato abbia cura di deallocare eventuali strutture dati allocate dinamicamente.

ESERCIZIO 2 (9 punti)

Si supponga di avere a disposizione ADT lista per interi (denominato come al solito `list`, con relative primitive). Si definisca una funzione ricorsiva:

```
list split(list l1)
```

che, ricevuta in ingresso la lista `l1` (contenente interi pari e dispari in un ordine qualsiasi), restituisca in uscita una lista contenente prima tutti i valori pari (in un qualunque ordine), e poi a seguire tutti i valori dispari (in un qualunque ordine). Ad esempio, se invocata con parametro `l1 = [0, 1, 2, 3, 4]`, la funzione può restituire come risultato la lista `[0, 2, 4, 3, 1]`. A tal scopo si utilizzino solo le operazioni primitive degli ADT `list`, e si definiscano, laddove necessario, altre funzioni ausiliarie.

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

char * run(char s[], int pos) {
      char * ch = s;
      int i;
      {
            int ch;
            for (i=0; i<pos; i++) {
                  ch = *(s+i);
                  printf("%c", ch); }
            printf("\n");
      }
      for (i=0; *ch != '\0'; ch++, i++);
      ch = (char *) malloc(sizeof(char)* (i-pos+1));
      for (; i>=pos; i--, ch[i-pos+1]=s[i+1]);
      return ch;
}

int main(void) {
      char s[] = "Paperino";
      char *p = run(s, 4);
      printf("%s\n", p);
      free(p);
      return (0);
}
```

ESERCIZIO 4 (5 punti)

Si consideri la seguente funzione:

```
int g(float a, float b) {
      a+=b;
      if (a <=0)
            return g(a, b) + g(a+1, b);
      else return b; }
```

Si dica che cosa restituisce se invocata con parametri **`g(-4.5, 2.0)`** e si mostrino i record di attivazione relativi.

ESERCIZIO 5 (3 punti) SOLO PER NUOVA MODALITÀ

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:

41-80

ESERCIZIO 1

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define DIM 32

typedef struct firma {
	char nome[DIM];
	char cognome[DIM];
} firma;

int confronta(firma firma1, firma firma2) {
	if	(	(strcmp(firma1.nome,firma2.nome) == 0 &&
			strcmp(firma1.cognome,firma2.cognome) == 0)
		||	(strcmp(firma1.nome,firma2.cognome) == 0 &&
			strcmp(firma1.cognome,firma2.nome) == 0) )
		return 1;
	else return 0; }

firma * eliminaDup(firma * l1, int dim1, firma * l2, int dim2, int * newDim) {
	int i, j, trovato;
	firma * result;
	result = (firma *) malloc(sizeof(firma) * (dim1+dim2));
	*newDim = 0;
	for (i=0; i<dim1; i++) {
		trovato = 0;
		for (j=0; j<*newDim && !trovato; j++)
			if	(confronta(l1[i],result[j])) trovato = 1;
		if (!trovato) {
			result[*newDim] = l1[i];
			*newDim = *newDim + 1;
		}
	}
	for (i=0; i<dim2; i++) {
		trovato = 0;
		for (j=0; j<*newDim && !trovato; j++)
			if ( confronta(l2[i], result[j])) trovato = 1;
		if (!trovato) {
			result[*newDim] = l2[i];
			*newDim = *newDim + 1; }
	}
	return result; }

int main(void) {
	FILE * fp1, *fp2;
	int dim1=0, dimTemp=0, dimResult=0, i;
	firma l1[100], * result = NULL, *temp;
	char filename[DIM];

	fp1 = fopen("files.txt", "r");
	while (fscanf(fp1, "%s", filename)!= EOF) {
		fp2 = fopen(filename, "r");
		temp = result;
		dimTemp = dimResult;
		dim1 = 0;
		while (fscanf(fp2, "%s %s", l1[dim1].cognome, l1[dim1].nome)!= EOF)
			dim1 = dim1 + 1;
		result = eliminaDup(l1, dim1, result, dimTemp, &dimResult);
		fclose(fp2);
		if (dimTemp >0) free(temp);
	}

	for (i=0; i<dimResult; i++) printf("%s %s\n", result[i].cognome, result[i].nome);
	free(result); fclose(fp1);
	return (0); }
```

ESERCIZIO 2

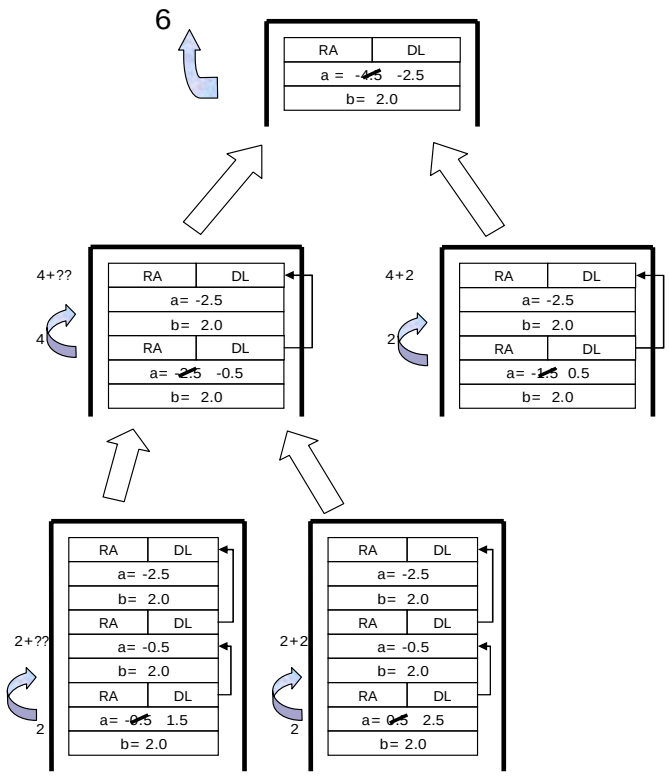
```
list append(list l1, list l2) {
       if (empty(l1)) return l2;
       else return cons(head(l1), append(tail(l1), l2));
}

list split(list l1) {
       if (empty(l1)) return emptylist();
       else if (head(l1)%2 == 0)
                     return cons(head(l1), split(tail(l1)) );
              else   return append(split(tail(l1)), cons(head(l1), emptylist()));
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
Pape
rino
La fuzione **main(…)** invoca la funzione **run(…)** con parametri "Paperino" e 4; questa funzione a sua volta provvede a stampare a video i primi 4 caratteri ("Pape",) quindi calcola quanto è lunga la stringa passata in ingresso, e alloca spazio sufficiente per contenere i restanti caratteri non ancora stampati (con anche spazio per il terminatore di stringa). In tale spazio vengono copiati (procedendo dall'ultimo fino al primo) i caratteri restanti, e la stringa così composta è restituita come risultato. Il programma principale si occupa poi di stamparla a video.

ESERCIZIO 4



ESERCIZIO 5

Sottrazione dei moduli (numero maggiore meno numero minore)

1010000-	80-
0101001=	41=
---	---
0100111	39

Aggiunta del segno
10100111