Fondamenti di Informatica L-A (A.A. 2006/2007) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Giovedì 13 Settembre 2007 – durata 2h

ESERCIZIO 1 (10 punti)

Data una struttura dati **parola** (da definire a cura del candidato) con due campi - una stringa di al più 30 caratteri e un intero rappresentante il numero di errori presenti in tale stringa -, il candidato realizzi la funzione

```
int simili(FILE* in, FILE* voc, parola *vett);
```

con parametri di ingresso due puntatori a file di testo già correttamente aperti in lettura (**in** e **voc**) ed un puntatore a un'area di memoria **vett** già allocata per contenere strutture dati **parola**.
Tale funzione deve restituire le stringhe contenute nel file **in** facenti parte del vocabolario passato in ingresso; in particolare tale funzione deve inserire nello spazio di memoria puntato da **vett** unicamente le stringhe che si discostano di al più 3 caratteri da una qualsiasi delle parole contenute nel file **voc**, come meglio spiegato in seguito. Se ad una parola di **in** corrispondono più parole che soddisfano tale condizione, il candidato deve inserire la distanza dalla parola di **voc** che si discosta meno.

A tale scopo si realizzi una funzione di supporto **compare** che, fornite in ingresso due stringhe ben formate, restituisca la distanza tra le due parole, confrontando uno a uno i caratteri delle due stringhe aventi lo stesso indice. Ad esempio, le stringhe **"ciao"** e **"cibo"** si discostano di un solo carattere (in quanto la distanza tra i caratteri **'a'** e **'b'** è 1), mentre **"acqua"** e **"aaqva"** di tre (2 per il secondo carattere e 1 per il quarto); nel caso in cui le due stringhe abbiano lunghezze diverse, il confronto termina non appena finisce una delle due stringhe.

La funzione **simili** deve restituire il numero di record **parola** inseriti nell'area di memoria **vett**.

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

ESERCIZIO 2 (9 punti)

Si supponga di avere a disposizione l'ADT lista per gli interi (denominato **list**, con relative primitive).
Si definisca una funzione **ricorsiva**:

```
list paridispari(list input, list* dispari);
```

che, ricevuta in ingresso la lista di interi **input**, restituisca per valore (come valore risultato) una lista contenente gli elementi di valore pari di **input** e per riferimento tramite il puntatore a lista **dispari** una lista di interi contenenti gli elementi di valore dispari di **input**.

ESERCIZIO 3 (7 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>

char* subset(char *i, int *dim, int start, int finish, float shift){
      char *temp;
      while(*i!='\0' && start>0){
            i++;
            start--;
            finish--;
      }
      temp=(char*)malloc(sizeof(char)*(finish));
      for(; finish>0 && *i!='\0'; i++){
            *(temp-start)=*i-(int)shift;
            finish--;
            start--;
      }
      *dim=-start;
      return temp;
}

int main(){
      char word[]="LaboRatoRio35h",*dest;
      int a,b,dim;   float f;
      a=11; b=18; f=2.7;
      dest=subset(word,&dim,a,b,f);
      printf("%s %d\n",dest,dim);
      return 0;
}
```

ESERCIZIO 4 (3 punti)
Si consideri la grammatica G con scopo S, simboli non terminali {X, Y, F, G} e simboli terminali {a,b,c,0,1,2}:

```
S::= GXY | FX
X::= FX | FYX
Y::= G | GYF
F::= 0 | 1 | 2
G::= a | b | c
```

La stringa **"12ac0b"** appartiene al linguaggio generato da tale grammatica?
In caso affermativo, se ne mostri la derivazione left-most.

ESERCIZIO 5 (3 punti)
È lecito scrivere

```
int* x;
int y = *x;
printf("%d\n",x);
printf("%d\n",y);
```

in questa precisa sequenza? Perché?
Che cosa compare a video?

ESERCIZIO 1

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

#define DIM 31

typedef struct{
	char stringa[DIM];
	int errori;
} parola;

int compare(char *w1, char *w2){
	int tot=0;
	while(*w1!='\0' && *w2!='\0'){
		int diff;
		diff=*w1-*w2;
		if(diff<0) diff=-diff;
		tot+=diff;
		w1++; w2++;
	}
	return tot;
}

int simili(FILE* in, FILE* voc, parola *vett){
	char word[DIM], voc_word[DIM]; int dim=0;
	while( (fscanf(in,"%s",word)!=EOF) ){
		int trovato=0;
		while( fscanf(voc,"%s",voc_word)!=EOF ){
			int diff=compare(word,voc_word);
			if(diff<=3){
				if(!trovato){
					trovato=1;
					strcpy((vett+dim)->stringa,word);
					(vett+dim)->errori=diff;
					dim++;
				}
				else{
					if(diff<(vett+dim-1)->errori){
						strcpy((vett+dim-1)->stringa,word);
						(vett+dim-1)->errori=diff;
					}
				}
			}
		}
		rewind(voc);
	}
	return dim;
}
```

ESERCIZIO 2

```
list paridispari(list input, list* dispari){
	if(empty(input))return emptylist();
	else if(head(input)%2==0){
		return cons(head(input), paridispari(tail(input),dispari));
	}
	else{
		*dispari=cons(head(input),*dispari);
		return paridispari(tail(input),dispari);
	}
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

```
13f 3
```

Nella funzione `main` viene creata una stringa ben formata inizializzata a `"LaboRatoRio35h"`, passata poi con altri argomenti alla funzione `subset`.
La funzione `subset` scorre l'array di un numero di caratteri uguale al valore presente in `start` o fino al carattere di terminazione; allo stesso tempo decrementa il valore di `finish`. In seguito la funzione `subset` alloca dinamicamente memoria sufficiente a contenere un numero di caratteri uguale al nuovo valore di `finish`. Il ciclo `for` continua ad iterare sull'array di caratteri ed inserisce nello spazio di memoria allocato dinamicamente il carattere attualmente preso in esame decrementato di due (il valore float 2.7 viene troncato a 2 tramite l'operazione di cast con `int`). Infine la funzione `subset` restituisce tramite `dim` il numero di caratteri inseriti nello spazio di memoria precedentemente allocato e restituisce per valore un riferimento alla memoria allocata.

ESERCIZIO 4

La stringa data inizia col simbolo terminale `'1'` e termina col simbolo terminale `'b'`.
Per poter ottenere una tale stringa, dovrebbe esserci una produzione con `F` come primo simbolo non terminale e `G` come ultimo simbolo non terminale, ma nessuna delle produzioni date ha tali caratteristiche. Infatti l'unica produzione che può terminare con `G` è `S::= GXY`, ma `G` come primo simbolo non terminale è incompatibile con la stringa data.

La stringa data non appartiene quindi al linguaggio generato dalla grammatica data.

ESERCIZIO 5

Il codice proposto non è corretto, in quanto la seconda riga assegna alla variabile `y` un valore non precedentemente inizializzato.
La prima `printf` stampa sullo standard output il valore del puntatore ad intero `x`, ovvero l'indirizzo a cui punta. La seconda `printf` stampa il valore presente nell'area di memoria puntata da `x`, ovvero un valore casuale (dato che tale area di memoria non è stata inizializzata).