Fondamenti di Informatica L-A (A.A. 2006/2007) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Giovedì 21 Giugno 2007 – durata 2h

ESERCIZIO 1 (9 punti)

Data una struttura di dati **`multa`** (da definire a cura del candidato in modo tale da contenere le sanzioni comminate durante un intero mese) con campi la targa del veicolo multato (un intero), il giorno del mese in cui è stata emessa la sanzione (un intero) e il suo ammontare (un float), il candidato realizzi la funzione

```
int calcola(multa* vMulte, int dimMulte, int* targhe, float* totale);
```

con parametri in ingresso un array di multe (**`vMulte`**) e la dimensione di tale vettore (**`dimMulte`**). Tale funzione ha lo scopo di calcolare per ogni veicolo (individuato dalla targa) l'ammontare totale da pagare (ovvero sommando per ciascun veicolo l'ammontare delle multe prese all'interno dell'intero mese in oggetto), considerando che per ciascun mese e per ciascun veicolo si è tenuti a pagare unicamente le prime quattro multe. In altre parole, la polizia municipale ha benevolmente deciso di richiedere il pagamento solo delle prime quattro multe per un dato veicolo in un dato mese.
La funzione **`calcola(...)`** deve restituire l'ammontare da pagare da ciascun veicolo tramite le aree di memoria indirizzate dai puntatori **`targhe`** e **`totale`** (opportunamente inizializzati dal chiamante prima dell'invocazione di **`calcola(...)`**), in modo tale che il valore presente in **`totale[i]`** sia da imputare al conducente del veicolo con targa **`targhe[i]`**. La funzione **`calcola(...)`** restituisca inoltre il numero di record presenti in tali aree di memoria.

ESERCIZIO 2 (9 punti)

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

```
list select(list l1, list l2)
```

che, ricevute in ingresso le liste **`l1`** e **`l2`**, restituisca in uscita una lista contenente gli elementi di **`l2`** corrispondenti al numero di caratteri alfabetici maiuscoli presenti negli elementi corrispondenti di **`l1`**. Se il numero di maiuscole nella prima stringa di **`l1`** è 0, allora il primo elemento della lista risultato sarà la prima stringa di **`l2`**, se è 1 la seconda stringa di **`l2`** e così via.
Ad esempio, date le liste **`l1=["trAttORe", "montagna", "Acqua", "MARE"]`** ed **`l2=["luce", "aria", "fiume", "lago", "atollo"]`** la lista restituita sarà **`["lago", "luce", "aria", "atollo"]`**, in quanto la stringa **`"trAttORe"`** contiene 3 maiuscole e va a selezionare la quarta stringa di **`l2`** ovvero **`"lago"`**, la stringa **`"montagna"`** contiene 0 maiuscole e porta a scegliere **`"luce"`**, e così via.

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>

char* shift(float* pos,char* str,int a,int b){
	char* res; int dim, i, index;
	dim=0;
	char ch;
	while(*str!='\0'){
		dim++;
		str++;
	}
	str=str-dim;
	res=(char*)malloc(sizeof(char)*dim);
	for(i=dim-1;i>-1;i--){
		index=pos[i];
		ch=str[index%dim];
		res[i]=ch;
		if(res[i]>=a && res[i]<=b)
			res[i]='-';
	}
	res[dim]='\0';
	return res;
}

int main(){
	float f[]={2.2, 1.0, 0.4, 1.8, 5.1, 3.0, 4.2, 6.7};
	char* s="ombrello";
	char* res;
	res=shift(f,s,'b','f');
	printf("%s\n",res);
	return 0;
}
```

ESERCIZIO 4 (5 punti)
Si consideri la seguente funzione:

```
char fun(float f, int i){
	i-=f;
	if(i>7) return 1+fun(f,i);
	else return 'a';
}
```

Si dica che cosa restituisce la funzione se invocata con parametri **`fun(2.1,17)`** e si mostrino i relativi record di attivazione.

ESERCIZIO 5 (3 punti)
Si mostri, tramite un esempio concreto di codice, come il linguaggio C permette al programmatore di gestire vettori la cui dimensione non sia già nota staticamente in fase di compilazione.
Nel caso in cui la dimensione dinamicamente decisa per il vettore suddetto non fosse sufficiente come si potrebbe rimediare?

ESERCIZIO 1

```
#include <stdio.h>

typedef struct{
	int targa;
	float valore;
	int giorno;
}multa;

int calcola(multa* vMulte,int dimMulte,int* targhe,float* totale){
	int i,j,dim,*flag;
	dim=0;
	flag=(int*)malloc(sizeof(int)*dimMulte);
	for(i=0;i<dimMulte;i++) flag[i]=0;

	for(i=0;i<dimMulte;i++){
		if(flag[i]==0){
			flag[i]=1;
			targhe[dim]=vMulte[i].targa;
			totale[dim]=vMulte[i].valore;
			int multeStessoMese=1;
			for(j=i+1;j<dimMulte;j++){
				if(vMulte[i].targa==vMulte[j].targa){
					multeStessoMese++;
					flag[j]=1;
					if(multeStessoMese<=4){
						totale[dim]=totale[dim]+vMulte[j].valore;
					}
				}
			}
			dim++;
		}
	}
	return dim;
}
```

ESERCIZIO 2

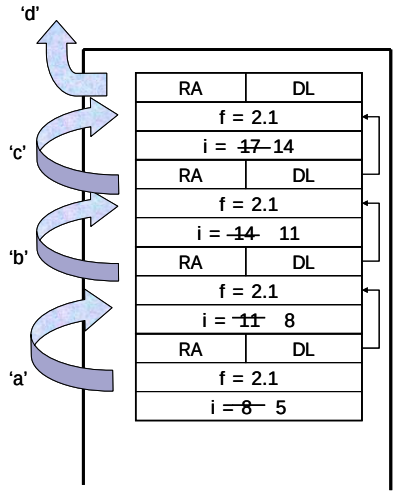
```
list select(list l1, list l2){
	char* str;   int maiuscole, i;  list temp;
	if(empty(l1)) return emptylist();
	else{
		maiuscole=0;
		str=head(l1);
		while(*str!='\0'){
			if(*str>='A' && *str<='Z')maiuscole++;
			str++;
		}
		temp=l2;
		for(i=0;i<maiuscole;i++){
			temp=tail(temp);
		}
		return cons(head(temp),select(tail(l1),l2));
	}
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
`-momlr-l`

Nella funzione main vengono creati un array di float ed una stringa ben formata, passati poi come argomenti alla funzione **`shift`**.
La funzione **`shift`** innanzitutto calcola la lunghezza della stringa ben formata **`str`**. In seguito la funzione **`shift`** alloca spazio sufficiente a contenere tutti i caratteri presenti nella stringa **`str`**. Il ciclo **`for`** inserisce caratteri nell'area di memoria appena allocata, dall'ultima locazione alla prima. In ciascuna posizione viene inserito il carattere dell'array **`str`** di indice **`pos[i]`** (troncato ad intero ed in modulo dimensione dell'array **`str`**). Il carattere ottenuto viene sostituito con '-' nel caso in cui sia compreso tra i caratteri 'b' ed 'f'. Viene poi restituito al chiamante un riferimento all'area di memoria precedentemente allocata.

ESERCIZIO 4



L'invocazione **`fun(2.1,17)`** restituisce il carattere 'd'.

ESERCIZIO 5

Per poter disporre di vettori di dimensione non nota staticamente in fase di compilazione è necessario allocare dinamicamente uno spazio di memoria di dimensioni adeguate utilizzando la funzione di libreria **`malloc`**. Ad esempio, per ottenere spazio sufficiente a contenere *X* interi è necessaria la seguente invocazione:

```
int *v1;    int X=5;
v1=(int*)malloc(sizeof(int)*X);
```

Lo spazio di memoria così allocato non è estendibile dinamicamente. Qualora fosse necessario disporre di uno spazio di memoria di dimensioni maggiori, ad esempio per contenere *Y* interi (*Y*>*X*), bisognerebbe effettuare una nuova allocazione dinamica e copiare i dati già presenti nel primo vettore nel secondo vettore appena allocato.

```
int *v2;    int i;    int Y=10;
v2=(int*)malloc(sizeof(int)*Y);
for(i=0;i<X;i++)
      v2[i]=v1[i];
```