Fondamenti di Informatica L-A (A.A. 2004/2005 - Ingegneria Informatica)
Prof.ssa Mello & Prof. Bellavista – Seconda Prova Intermedia del 02/12/2004 - durata 2h
COMPITO B

ESERCIZIO 1 (13 punti)

Una società di assicurazioni per autisti gestisce un programma di premiazione per "autisti prudenti". In particolare, per ogni assicurato viene salvato su un file binario "**`risarcimenti.dat`**" il nome dell'autista (al massimo 61 caratteri), e un numero (di tipo **`float`**) che rappresenta l'ammontare di euro di cui è stato risarcito dalla società il tale autista. Tali informazioni sono organizzate come una struttura **`driver`**, opportunamente definita dal candidato.
Si scriva una funzione:

```
int readMoney (char driversFile[], driver results[], int maxDim, float maxMoney)
```

che, ricevuto in ingresso il nome di un file **`driversFile`**, un array **`results`** di strutture **`driver`**, la dimensione massima dell'array **`maxDim`**, e un limite superiore di denaro **`maxMoney`**, copi nell'array **`results`** i dati dei clienti che hanno almeno un risarcimento totale minore o uguale di **`maxMoney`**. La funzione deve restituire come risultato il numero di utenti che hanno incassato al massimo **`maxMoney`**; si noti che tale risultato rappresenta anche la dimensione logica dell'array **`results`**. Qualora il file non sia accessibile, la funzione deve restituire il valore -1 **(6 punti)**.

Si scriva poi un programma **`main()`** che chieda all'utente il numero di assicurati salvati sul file (tale numero sarà noto solo a tempo di esecuzione), e allochi dinamicamente un vettore **`V`** di **`driver`** sufficientemente grande per poter contenere, nel caso peggiore, i dati di tutti gli autisti salvati in **`driversFile`**. Il programma dovrà poi chiedere all'utente la soglia superiore in denaro e, utilizzando la funzione **`readMoney()`**, leggere da file e memorizzare in **`V`** i dati degli utenti che hanno ottenuto risarcimenti per un valore al massimo pari a **`maxMoney`**.
Il programma infine deve stampare a video il nome ed il risarcimento degli assicurati contenuti in **`V`** se e solo se il loro nome comincia per "Be" **(7 punti).**

ESERCIZIO 2 (10 punti)

Si scriva una funzione **`leggi()`** che faccia inserire da console una serie di numeri interi (positivi e negativi) dispari e li memorizzi in ordine crescente in una lista. Qualora l'utente inserisca un numero pari, tale numero deve essere semplicemente scartato. L'utente può segnalare la fine della fase di inserimento numeri digitando il valore 0. Si supponga di possedere il tipo di dato astratto **`list`**, con le operazioni primitive associate e le addizionali operazioni **`list insord(element e, list l)`** e **`int member(element e, list l)`** viste a lezione. Tali funzioni possono anche non essere riportate nella soluzione.
Dopo aver caricato i valori nella lista, la funzione **`leggi()`** deve creare una nuova lista contenente i valori letti precedentemente ma senza ripetizioni e deve restituire questa ultima lista come risultato finale.

Si scriva poi una funzione **`stampaLista(list l)`** che, utilizzando la notazione a puntatori (e quindi non utilizzando operazioni primitive), stampi tutti gli elementi multipli di 3 contenuti nella lista **`l`**.

ESERCIZIO 3 (6 punti)

Nel caso in cui il programma sorgente C seguente compili ed esegua correttamente, se ne indichino i valori stampati a tempo di esecuzione, motivando la risposta data. In caso di errori di compilazione o errori runtime, si descriva invece nel dettaglio la motivazione di tali errori.
Si indichino inoltre quali sono i blocchi in cui sono visibili le variabili **N** e **L**, specificando se si tratta di variabili locali o globali. Si motivi opportunamente la risposta.

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

int L = 9;
int N = 4;

void write(char * v, char * w, int start, int end) {

    while ((start < end) && (*(w+start)!='\0')) { // versione corretta…
      *(v+start) = *(w+start);
      start++;
    };
    *(v+start) = '\0';
    return;
}

int main () {
    char temp[] = "Dolomiti";
    char * name;
    int N=L;
    int  i;

    name = (char *) malloc((N) * sizeof(char));
    write(name, temp, 0, 17);

    printf("Adesso L vale: %d\n", L);

    for (i=0; (i<N && (*(name+i)!= '\0')); i++)
      printf("%c", *(name+i));

    return 0;
}
```

ESERCIZIO 4 (3 punti)

Si presenti brevemente come il linguaggio C effettua il passaggio di parametri nella chiamata a funzione/procedura. In particolare, si illustri che cosa succede nei casi concreti di passaggio di una struct e di passaggio di un puntatore a float come parametri di ingresso.

Soluzione Compito B

Esercizio 1

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

#define DIM 62

typedef struct {
    char name[DIM];
    float money;
} driver;

int readMoney (char driversFile[], driver results[], int maxDim, float maxMoney) {
    FILE * f;
    int i;
    int logicDim = 0;

    f = fopen(driversFile, "rb");
    if (f == NULL) return(-1);

    for (i=0;     i<maxDim &&
                  (fread( &results[logicDim], sizeof(driver), 1, f) > 0); i++) {
      if (results[logicDim].money <= maxMoney)
            logicDim++;
    }

    fclose(f);
    return logicDim;
}

int main() {

    driver * V;
    int i, maxUtenti, logicDim;
    float maxMoney;

    printf("Inserire numero massimo di utenti da leggere: ");
    scanf("%d", &maxUtenti);
    V = (driver *) malloc(sizeof(driver) * maxUtenti);

    printf("Inserire risarcimento massimo: ");
    scanf("%f", &maxMoney);

    logicDim = readMoney("risarcimenti.dat", V, maxUtenti, maxMoney);
    if (logicDim < 0)
      exit(-1);

    for (i=0;i<logicDim; i++)
      if ((V[i].name[0] == 'B') && (V[i].name[1] == 'e'))
            printf("Utente: %s, euro %6.2f\n", V[i].name, V[i].money);
    free(V);
    return 0;
}
```

Esercizio 2

```
list leggi() {
	int temp;

	list a, result;

	a = emptylist();
	result = emptylist();
	do {
		printf("Inserire numero: ");
		scanf("%d", &temp);
		if ((temp%2) == 1)
			a = insord(temp, a);
	} while (temp != 0);

	while (!empty(a)) {
		if (!member(head(a), result) )
			result = insord(head(a), result);
		a = tail(a);
	}
	return result;
}

void stampaLista(list l) {
	if (l==NULL)
		return;
	else {
		if ((l->value %3) == 0)
			printf("%d\n", l->value);
		stampaLista(l->next);
	}
}
```

Esercizio 3

Il programma stampa:

```
Adesso L vale: 9
Dolomiti
```

Il programma **main** dichiara un array **temp** di caratteri, subito inizializzato come stringa a “Dolomiti”. Quindi **temp** avrà dimensione 9 (per via del terminatore). Poi viene allocata dinamicamente della memoria, assegnata al puntatore **name**. In particolare, viene allocato spazio per 9 caratteri (infatti la variabile **N**, locale al **main**, è stata inizializzata con il valore di **L**, cioè 9).
Viene invocata la funzione **write**, che copia i caratteri memorizzati in **w**, nel vettore di destinazione **v**, a partire dalla posizione **start** del vettore **w** nella corrispondente posizione di **v**, fino alla posizione **end** (esclusa). La funzione si ferma in caso il vettore **w** contenga il terminatore di stringa prima della posizione **end**. Al termine del ciclo comunque in **v** viene copiato anche un terminatore di stringa.
Infine il programma main si occupa di stampare il valore di **L** (che vale 9) e poi il contenuto della zona di memoria puntata da **name**; tale area è stata “riempita”dalla funzione **write**, e quindi viene stampato “Dolomiti”.