Fondamenti di Informatica L-A (A.A. 2002/2003) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Seconda Prova Intermedia del 09/12/2002 - durata 2h
COMPITO A

ESERCIZIO 1 (12 punti)

Si scriva una funzione che riceva come parametri di ingresso un file di testo **F** di strutture (persona) composte da un campo cognome (stringa di al più 20 caratteri) e da un campo valore (intero) e restituisca come parametro di uscita un vettore **Vnew** contenente le strutture di **F** il cui campo valore è compreso fra un valore **Min** e un valore **Max** ed il loro numero **N** (6 punti).

```
void positivi(FILE * F, persona Vnew[], int Min, int Max, int *N);
```

È dato un file di testo **risultati.txt**, contenenti i risultati dell'esame di Fondamenti di Informatica. In particolare, si supponga che il file sia costituito da righe, ciascuna contenente una stringa (cognome dello studente) e un intero (che rappresenta il voto conseguito nella prova intermedia), separati da uno o più spazi (stessi campi della struttura persona sopra definita). Ad esempio, **risultati.txt**:

```
ROSSI      28
BIANCHI    15
ALBERTI    30
MONARI     19
.....
```

Si scriva un programma C che inserisca in un vettore **v1** (supposto di dimensione massima **DIM 100**) tutti gli studenti (cognome e voto) contenuti nel file **risultati.txt** che hanno conseguito risultati compresi fra **18 e 30** richiamando la procedura **positivi()** precedentemente definita. Inoltre, il programma stampi il contenuto di **v1** a terminale (6 punti).

ESERCIZIO 2 (6 punti)

Si scriva una funzione ricorsiva **inc()** che data in ingresso una lista di interi, restituisca in uscita una lista in cui gli elementi pari sono incrementati di uno e gli elementi dispari sono decrementati di uno.

Si supponga di possedere il tipo di dato astratto **list** e le sue operazioni PRIMITIVE (che quindi possono NON essere riportate nella soluzione).

ESERCIZIO 3 (10 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.

Indicare inoltre quali sono i blocchi in cui è visibile la variabile N, sempre motivando la risposta.

```
#include <stdio.h>
#include <stdlib.h>
#define L 10

int N=L;

void P(float y[], int DIM) {
int i;
N++;
for (i=0; i<DIM-1; i=i+N/5) y[i]++;
}

main () {
float* v;
int i;

v = (float *) malloc(L*sizeof(float));
for (i=0; i<L; i++) v[i]=2*i+2;

if (v[1]) P(v,L-2);

{ int N;
  N=N-2;
}
printf("N vale adesso: %d\n",N);

for (i=0; (i=i+2)<N-1; ) printf("%f\n", v[i] + *(v+i+1));
}
```

Esercizio 4 (4 punti)

Quale è la differenza fra la variabile V e la variabile **punt** definite nel codice seguente? (2 punti)

```
int V[10]; int *punt;
punt=(int*) malloc(10*sizeof(int)) ;
```

La seguente istruzione è lecita e, in caso affermativo, che cosa produce? Se ne motivi opportunamente la risposta (2 punti).

```
V=punt;
```

Fondamenti di Informatica L-A (A.A. 2002/2003) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Seconda Prova Intermedia del 09/12/2002 - durata 2h
COMPITO B

ESERCIZIO 1 (12 punti)

Si scriva una funzione che riceva come parametri di ingresso un file di testo **F** di strutture (merce) composte da un campo nome (stringa di al più 100 caratteri) e un campo quantità (intero) e restituisca come parametro di uscita un vettore **Vnew** contenente le strutture di **F** il cui campo quantità è diverso da entrambi i valori **N1** e **N2** ed il loro numero **N** (6 punti).

```
void scelti(FILE * F, merce Vnew[], int N1, int N2, int *N);
```

È dato un file di testo **magazzino.txt**, contenente le merci di un certo magazzino di cartoleria. In particolare, si supponga che il file sia costituito da righe ciascuna contenente una stringa (nome del materiale) e un intero (che rappresenta la quantità), separati da uno o più spazi (stessi campi della struttura merce sopra definita). Ad esempio, **magazzino.txt**:

```
LIBRI      280  
QUADERNI  150  
MATITE    500  
.....
```

Si scriva un programma C che inserisca in un vettore **V1** (supposto di dimensione massima **DIM 200**) tutte le merci (nome e quantità) contenute nel file **magazzino.txt** che hanno una quantità diversa da 0 e 1, chiamando la funzione **scelti()** precedentemente definita. Inoltre, il programma stampi il contenuto di **V1** a terminale (6 punti).

ESERCIZIO 2 (6 punti)

Si scriva una funzione ricorsiva **aggiorna()** che data in ingresso una lista di interi ed un numero **num**, restituisca in uscita una lista in cui gli elementi uguali a **num** sono posti uguali a zero e gli altri (diversi da **num**) sono posti uguali a 1.

Si supponga di possedere il tipo di dato astratto **list** e le sue operazioni PRIMITIVE (che quindi possono NON essere riportate nella soluzione).

ESERCIZIO 3 (10 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.

Indicare inoltre quali sono i blocchi in cui è visibile la variabile N, sempre motivando la risposta.

```
#include <stdio.h>
#include <stdlib.h>
#define L 10

int N=L;

void Q(float y[], int DIM) {
int i;
N++;
for (i=0; i<DIM-1; i=i+3) y[i]=y[i]*2;
}

main () {
float* v;
int i;

v = (float *) malloc(L*sizeof(float));
for (i=0; i<L; i++) v[i]=2*i+1;

if (v[2]) Q(v,L-2);

{ int N;
  N=N*7;
}
printf("N vale adesso: %d\n",N);

for (i=-3; (i=i+3)<N-2; ) printf("%f\n",(*(v+i+1)) + v[i]);
}
```

ESERCIZIO 4 (4 punti)

Quale è la differenza fra la variabile **s1** e la variabile **s2** definite nel codice seguente? (2 punti)

```
char s1[]="Pippo", *s2;
s2=(char*) malloc(strlen("Pippo")+1) ;
```

La seguente istruzione è lecita e, in caso affermativo, che cosa produce? Se ne motivi opportunamente la risposta (2 punti).

```
s2=s1;
```

Soluzione Compito A

Esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 100

typedef struct {char cognome[21]; int valore;} persona;

void positivi(FILE * F, persona Vnew[], int Min, int Max, int *N) {
    persona per; *N=0;

    while(fscanf(F,"%s %d\n", per.cognome, &per.valore) != EOF) {
        if ((per.valore <= Max )&& (per.valore >= Min))
            { Vnew[*N]=per; *N= ++(*N);}
    }
}

main() {
    persona V1[DIM];
    int Num, i=0;
    FILE* f;

    if ((f=fopen("risultati.txt", "r"))==NULL) {
        printf("Il file non esiste!"); exit(1); }
    positivi(f, V1, 18, 30, &Num);
    for (i=0; i<Num; i++)
        printf("%s %d\n", V1[i].cognome, V1[i].valore);
    fclose(f);
}
```

Esercizio 2

```
list inc(list l) {

if empty(l) return emptylist();
else if (head(l)%2) cons(head(l)-1, inc(tail(l)));
    else cons(head(l)+1, inc(tail(l)));
}
```

Esercizio 3

Il programma stampa:

N vale adesso: 11

15.000000

23.000000

31.000000

38.000000

Infatti, la procedura P() viene invocata con passaggio per riferimento del vettore allocato dinamicamente e va a incrementare di 1 tutti gli elementi del vettore di posizione pari compresa fra 0 e 6, estremi inclusi. Il programma principale stampa il valore di N variabile globale incrementata a 11 in P() e poi il risultato della somme degli elementi consecutivi del vettore ($v[2]+v[3]$, $v[4]+v[5]$, $v[6]+v[7]$, $v[8]+v[9]$).

La variabile N è visibile in tutti i blocchi, perché è definita nell'ambiente globale. Esiste però un ulteriore blocco all'interno del main (dopo if e prima di printf) in cui è definita una nuova variabile intera con lo stesso nome che, all'interno di quel blocco, copre la variabile globale N e ne impedisce la visibilità.

Esercizio 4

Allocazione statica da una parte (vettore V), allocazione dinamica dall'altra (area puntata da punt).

Inoltre non è mai possibile in C riassegnare l'indirizzo di memoria mantenuto in una variabile di tipo array come V. Per questo l'assegnamento **V=punt;** mostrato nel codice è errato e produce un errore di compilazione.

Soluzione Compito B

Esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 200

typedef struct {char nome[101]; int quantita;} merce;

void scelti(FILE * F, merce Vnew[], int N1, int N2, int *N) {
    merce m; *N=0;

    while(fscanf(F,"%s %d\n", m.nome, &m.quantita) != EOF) {
        if ((m.quantita != N1) && (m.quantita != N2))
            { Vnew[*N]=m; *N= ++(*N);}
    }
}

main() {
    persona V1[DIM];
    int Num, i=0;
    FILE* f;

    if ((f=fopen("magazzino.txt", "r"))==NULL) {
        printf("Il file non esiste!"); exit(1); }
    scelti(f, V1, 0, 1, &Num);
    for (i=0; i<Num; i++)
        printf("%s %d\n", V1[i].nome, V1[i].quantita);
    fclose(f);
}
```

Esercizio 2

```
list aggiorna(list l, int num) {

if empty(l) return emptylist();
else if (head(l)==num) cons(0, aggiorna(tail(l),num));
    else cons(1, aggiorna(tail(l),num));
}
```

Esercizio 3

Il programma stampa:

N vale adesso: 11

5.000000

23.000000

41.000000

Infatti, la procedura `Q()` viene invocata con passaggio per riferimento del vettore allocato dinamicamente e va a raddoppiare gli elementi del vettore di posizione 0, 3, e 6. Il programma principale stampa il valore di `N` variabile globale incrementata a 11 in `Q()` e poi il risultato della somme degli elementi consecutivi del vettore (`v[1]+v[0]`, `v[4]+v[3]`, `v[7]+v[6]`).

La variabile `N` è visibile in tutti i blocchi, perché è definita nell'ambiente globale. Esiste però un ulteriore blocco all'interno del `main` (dopo `if` e prima di `printf`) in cui è definita una nuova variabile intera con lo stesso nome che, all'interno di quel blocco, copre la variabile globale `N` e ne impedisce la visibilità.

Esercizio 4

Allocazione statica da una parte (vettore `s1`), allocazione dinamica dall'altra (area puntata da `s2`).

Inoltre non è mai possibile in C riassegnare l'indirizzo di memoria mantenuto in una variabile di tipo array come `s1`. L'assegnamento `s2=s1`; mostrato nel codice è corretto. Dopo l'assegnamento `s2` punta alla stessa area di memoria in cui è stata allocata staticamente la stringa "Pippo", riferibile anche attraverso il puntatore `s1`. Viene perso definitivamente il riferimento all'area di memoria allocata dinamicamente tramite `malloc()`.