Fondamenti di Informatica L-A (A.A. precedenti al 2008/2009) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista
Prova d'Esame di Mercoledì 13 Gennaio 2010 – durata 2h

ESERCIZIO 1 (10 punti)

Per informatizzare la gestione di un magazzino un'azienda registra in un file di testo **`magazzino.txt`** l'elenco di tutti prodotti che sono stati venduti o resi giorno per giorno in un dato mese. Ciascuna riga è formata dal nome del prodotto (al più 10 caratteri), uno spazio, un intero rappresentante il numero di prodotti venduti (intero negativo) o resi (intero positivo) ed un intero rappresentante il giorno del mese a cui si riferisce la riga (ad esempio 1 per il primo giorno del mese, 2 per il secondo e così via).

Dopo aver definito un'opportuna struttura dati **`oggetto`** con lo scopo di rappresentare le informazioni relative ad un prodotto (nome oggetto + resi/venduti), il candidato realizzi le seguenti funzioni:

a) `int prodotti(FILE* fp, int primo, int ultimo);`

che, ricevuti in ingresso un puntatore a file e due interi rappresentanti giorno iniziale e giorno finale, in base alle informazioni contenute nel file dato restituisca il numero di prodotti che sono stati gestiti nel lasso di tempo compreso tra **`primo`** ed **`ultimo`** compresi, ovvero il numero di prodotti che sono stati venduti o resi almeno una volta. ATTENZIONE: è noto che nel magazzino esistono al più 100 prodotti.

b) `oggetto* magazzino(char *nomeFile, int primo, int ultimo, int *dim);`

che, ricevuti in ingresso il nome di un file di testo e due interi rappresentanti giorno iniziale e giorno finale, restituisca un array di strutture **`oggetto`** (e relativa dimensione) contenente tutti i prodotti gestiti nel lasso di tempo indicato, avendo cura di specificare in ciascun elemento dell'array restituito una sintesi dello stato del magazzino nel lasso di tempo indicato. Ad esempio, se l'oggetto **`acqua`** è stato venduto 5 volte e reso 2 nel lasso di tempo indicato, l'elemento corrispondente dovrà contenere il valore -3. ATTENZIONE: la dimensione dell'array restituito deve essere la più piccola possibile; a tal scopo si consiglia di sfruttare la funzione realizzata al punto precedente.

Si realizzi infine una funzione **`main()`** che invochi correttamente le funzioni di cui ai punti a) e b).

ESERCIZIO 2 (10 punti)

Si supponga di avere a disposizione l'ADT lista definito per le stringhe ben formate.

Il candidato definisca la funzione *ricorsiva*

`int words(list stringList, char* substring);`

che restituisce il numero di elementi della lista **`stringList`** che iniziano esattamente con i caratteri della stringa ben formata **`substring`**. Ad esempio, se **`stringList = { "casa", "mare", "montagna", "maestro" }`**, e **`substring = "ma"`**, la funzione **`words(...)`** restituisce 2 dato che **`"mare"`** e **`"maestro"`** iniziano per **`"ma"`**. La funzione **`words`** deve essere implementata utilizzando le sole primitive dell'ADT lista.

A tal scopo si realizzi e si utilizzi la funzione di supporto

`int startsWith(char* string, char *subs);`

che, date due stringhe ben formate **`string`** e **`subs`**, restituisce 1 se la stringa **`string`** inizia esattamente coi caratteri della stringa **`subs`**, 0 altrimenti. Per ipotesi la stringa ben formata **`subs`** non è mai più lunga della stringa ben formata **`string`**.

ESERCIZIO 3 (3 punti)

Si consideri la seguente funzione:

```
int fun(float x, int y){
      int z;
      z=x*y;
      if( (y%3)==1)
            return 0;
      else
            return x + fun(x, z);
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (3.7, 3.7).

ESERCIZIO 4 (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 <stdlib.h>

int* create(int* a1, int* a2, int dim1, int *dim2){
    int* b, *temp;
    b=(int*)malloc(sizeof(int)*dim1**dim2);
    temp=b;
    while(*dim2>0 && (*a1)!=(*a2)){
        if( ((*dim2)%2)==0 ){
            *b=*a1;
            a1++;
        }
        else{
            *b=*a2;
            a2--;
            (*dim2)--;
        }
        b++;
    }
    *dim2=b-temp;
    return temp;
}

int main(){
    int *res, i;
    int dim2=3;
    int a[]={1,2,3,4,5,6,7};
    int b[]={4,5,6};
    res=create(a, b+dim2-1, 7, &dim2);
    for(i=dim2-1; i>=0; i--){
        printf("%d\n", res[dim2-i-1]);
    }
    return 0;
}
```

ESERCIZIO 5 (3 punto)

Illustrare sinteticamente la differenza fra approccio compilato, interpretato ed intermedio.

Soluzioni

ESERCIZIO 1

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

#define DIM 100

typedef struct{
    char nome[11];
    int resi;
} oggetto;

int prodotti(FILE* fp, int primo, int ultimo){
    oggetto letti[DIM];
    int giorno, res, i, trovato;
    res=0;
    while( fscanf(fp,"%s %d %d", letti[res].nome, &(letti[res].resi),
&giorno)==3 ){
        if(giorno>=primo && giorno<=ultimo){
            trovato=0;
            for(i=0; trovato==0 && i<res; i++){
                if( strcmp(letti[i].nome,letti[res].nome)==0 ){
                    trovato=1;
                }
            }
            if(!trovato){
                res++;
            }
        }
    }
    return res;
}

oggetto* magazzino(char *nomeFile, int primo, int ultimo, int *dim){
    oggetto *res;
    FILE *fp;
    char nome[11];
    int venduti, giorno, trovato, i, count;
    fp = fopen(nomeFile, "rt");

    if (fp == NULL) {
            printf("Errore aprendo il file %s\n", nomeFile);
            system("PAUSE");
            exit(-1);
    }

    *dim=prodotti(fp,primo,ultimo);
    res=(oggetto*)malloc(sizeof(oggetto)**dim);
    rewind(fp);

    count=0;
    while( fscanf(fp,"%s %d %d", nome, &venduti, &giorno)==3 ){
```

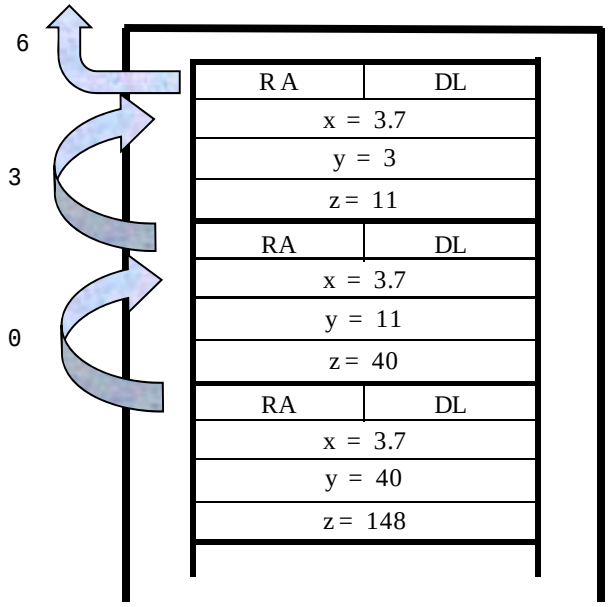
```
        if(giorno>=primo && giorno<=ultimo){
            trovato=0;
            for(i=0; trovato==0 && i<count; i++){
                if( strcmp(res[i].nome,nome)==0 ){
                    res[i].resi+=venduti;
                    trovato=1;
                }
            }
            if(!trovato){
                strcpy(res[count].nome,nome);
                res[count].resi=venduti;
                count++;
            }
        }
    }

    fclose(fp);
    return res;
}

int main(){
    int dim, i;
    oggetto *array;
    array = magazzino("magazzino.txt", 1, 8, &dim);
    for(i=0; i<dim; i++){
        printf("%s %d\n", array[i].nome, array[i].resi);
    }
    free(array);
    return 0;
}
```

ESERCIZIO 2

```
int startsWith(char* string, char *subs){
    int ok=1;
    // per ipotesi string sempre più lunga di subs
    while(ok && (*subs!='\0') ){
        if(*string!=*subs){
            ok=0;
        }
        string++; subs++;
    }
    return ok;
}
int words(list l, char* substring){
    char* s;
    if(empty(l)){
        return 0;
    }
    else{
        s=head(l);
        return startsWith(head(l), substring) + words(tail(l),substring);
    }
}
```

ESERCIZIO 3

La funzione restituisce il valore 6.



ESERCIZIO 4

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

```
6
1
2
3
4
```

La funzione `main()` invoca la funzione `create()` con parametri attuali rispettivamente l'indirizzo della prima cella di memoria dell'array `a`, l'indirizzo dell'ultima cella di memoria dell'array `b`, l'intero 7 e l'indirizzo della variabile `dim2`.

La funzione `create()` alloca spazio sufficiente a contenere 21 interi. Sia la variabile `b` che la variabile `temp` della funzione `create()` puntano alla prima cella della memoria allocata dinamicamente. Notare che la variabile `b` della funzione `create()` non ha alcuna relazione con la variabile `b` della funzione `main()`.
La funzione procede ad assegnare alle celle di memoria appena allocate il valore delle celle degli array `a1` (se il valore referenziato da `dim2` è pari) ed `a2` (se il valore referenziato da `dim2` è dispari), scorrendo `a1` dall'inizio verso la fine ed `a2` dalla fine verso l'inizio. Infine la funzione `create()` assegna alla cella di memoria referenziata da `dim2` il valore corrispondente al numero di celle dell'array `b` correttamente inizializzate e restituisce al chiamante l'indirizzo della prima cella di memoria allocata dinamicamente.

La funzione `main()` infine stampa a video (dalla prima all'ultima) le celle di memoria correttamente inizializzate dalla funzione `create()`.