Fondamenti di Informatica TL-A (A.A. 2008/2009) - Ingegneria Informatica
Prof.ssa Mello
Prova d'Esame di Venerdì 19 Giugno 2009 – durata 2h (a.a. precedenti)

ESERCIZIO 1 (12 punti)

Un'agenzia turistica memorizza in un file di testo l'elenco degli **`Hotel`** convenzionati. Ogni **`Hotel`** è caratterizzato da un nome (stringa di al più 20 caratteri, senza spazi), un numero di stelle (intero), una tipologia (carattere uguale a 'M' in caso di hotel in montagna, 'S' in caso di hotel al mare), una capienza (intero) e il numero di posti attualmente disponibili. All'interno del file ogni riga è associata ad un hotel, e i vari campi sono memorizzati nell'ordine appena descritto, separati da spazi. Un esempio di file degli hotel potrebbe essere:

```
miramonti 3 M 20 12
marebello 2 S 10 7
belvedere 2 M 15 0
```

Si realizzi un programma che acquisisca l'insieme degli hotel convenzionati da file di testo, e fornisca una funzione per filtrare l'insieme degli hotel selezionando solo quelli che rispondono a un certo criterio.

In particolare, dopo aver definito l'ADT **`Hotel`** , si realizzi una funzione

```
Hotel* leggiHotels(FILE* fp, int *dim);
```

che riceva in input un puntatore al file di testo contenente le informazioni sugli hotel e provveda a leggere tali informazioni, memorizzandole in un vettore di **`Hotel`** allocato dinamicamente della dimensione strettamente necessaria; al termine della funzione, **`dim`** deve puntare alla dimensione logica del vettore.

Si realizzi poi una funzione

```
Hotel* filtro(Hotel* hotelArray, int dim, int minStelle, char tipo, int* dimFiltro);
```

che riceva in input un vettore di hotel con la sua dimensione logica, il numero minimo di stelle desiderato dall'utente nonchè il tipo richiesto, e provveda a restituire in output un vettore, allocato dinamicamente (della stessa dimensione del vettore preso in input), che contenga solo gli hotel che rispettano il seguente criterio:

- Presenza di almeno un posto disponibile
- Numero di stelle superiore o uguale a **`minStelle`**
- Tipologia corrispondente a **`tipo`** (se **`tipo=='A'`** , qualunque tipologia va bene)

Al termine della funzione, **`dimFiltro`** deve puntare alla dimensione logica del vettore contenente gli hotel selezionati.

Si realizzi infine un **`main(…)`** in cui si utilizzano le funzioni precedentemente definite per acquisire un insieme di hotel dal file **`hotels.txt`**, selezionando e stampando a video nome, stelle e posti disponibili degli hotel di montagna con almeno due stelle.
Funzioni di utilità: **`rewind(…)`** per riportare il file pointer all'inizio del file.

ESERCIZIO 2 (10 punti)

Si scriva una funzione

```
float mediaPesata(list voti, list crediti);
```

che riceve in ingresso i puntatori **`voti`** e **`crediti`** alle radici di due liste contenenti una serie di interi. La funzione deve innanzitutto verificare che le due liste abbiano la stessa lunghezza; a tal scopo, si supponga di avere a disposizione una funzione **`int lunghezza(list l)`** che calcoli la lunghezza di una lista. Nel caso in cui le due liste abbiano la stessa lunghezza, la funzione deve restituire la media dei voti pesata rispetto ai crediti; in caso contrario, la funzione deve restituire un valore di errore. Si faccia l'ipotesi che ogni coppia voto/crediti sia costituita da elementi presenti nelle due liste alla stessa posizione.
Ad esempio, supponendo di avere una lista di voti [30,26,28] e una lista di crediti [6,6,3] la funzione deve

restituire il valore:

$$28.4 = \frac{30 \cdot 6 + 28 \cdot 6 + 26 \cdot 3}{6 + 6 + 3}$$

La realizzazione della funzione si deve appoggiare su due ulteriori funzioni ausiliarie:

1. Una funzione iterativa

   ```
   int somma(list l)
   ```

   che riceve in ingresso una lista di interi e ne restituisce la somma. Per questa funzione non viene data a disposizione nessuna libreria predefinita sulle liste.

2. Una funzione ricorsiva

   ```
   float sommaPesata(list valori, list pesi);
   ```

   che riceve in ingresso una lista di valori e una lista di pesi e restituisce la somma dei valori, ognuno moltiplicato per il corrispondente peso. Per questa funzione si utilizzino le funzioni di libreria predefinite sulle liste di interi.

ESERCIZIO 3 (5 punti)

Si consideri la seguente funzione:

```
int tar(int r, int c)
{
    if(r < 1 || c < 1 || r < c)
        return 0;
    else {
        if(r == 1)
            return 1;
        else
            return tar(r-1,c-1) + tar(r-1,c);
    }
}
```

Mostrare la sequenza dei record di attivazione nel caso in cui la funzione sia invocata con parametri attuali (3,2) e il valore di ritorno.

ESERCIZIO 4 (3 punti)

Si rappresenti schematicamente l'architettura di una macchina di Von Neumann e ne si descrivano i componenti.

ESERCIZIO 5 (2 punti)

Si consideri la grammatica G con scopo S, simboli non terminali {A, B, C, D, X} e simboli terminali {f,g,m,n,1,2,3}:

```
S::= ABA | BAB
A::= X | XC
B::= X | DXD
C::= f | g
D::= m | n
X::= XB | 1 | 2 | 3
```

La stringa **"mn11f3g"** appartiene al linguaggio generato da tale grammatica? In caso affermativo, se ne mostri la derivazione left-most.

Soluzioni

ESERCIZIO 1

```
typedef struct
{
      char nome[21];
      int stelle;
      int capienza;
      int postiLiberi;
      char tipo;
} Hotel;

boolean leggiHotel(FILE* fp, Hotel* h)
{
      int read = fscanf(fp, "%s %d %c %d %d",    h->nome,
                                                 &h->stelle,
                                                 &h->tipo,
                                                 &h->capienza,
                                                 &h->postiDisponibili
                                                 );
      return read == 5;
}

Hotel* leggiHotels(FILE* fp, int *dim)
{
      int i;
      Hotel h;
      Hotel* hotelArray;
      *dim = 0;
      while(leggiHotel(fp,&h))
            (*dim)++;
      rewind(fp);
      hotelArray = (Hotel*)malloc(*dim*sizeof(Hotel));
      for(i = 0; i < *dim; i++)
            leggiHotel(fp,&(hotelArray[i]));
      return hotelArray;
}

Hotel* filtro(Hotel* hotelArray, int dim, int minStelle, char tipo, int* dimFiltro)
{
      int i;
      Hotel* hotelArrayFiltrato = (Hotel*)malloc(dim*sizeof(Hotel));
      *dimFiltro = 0;
      for(i = 0; i < dim; i++)
      {
            if(  hotelArray[i].postiLiberi > 0
                  && hotelArray[i].stelle >= minStelle
                  && (tipo=='A' || tipo == hotelArray[i].tipo))
                        hotelArrayFiltrato[(*dimFiltro)++] = hotelArray[i];
      }
      return hotelArrayFiltrato;
}

int main() {
```

```
      FILE* fpDB;

      Hotel* hotelArray;
      Hotel* hotelArrayFiltrato;
      int dim, dimFiltro, i;
      fpDB = fopen("hotels.txt","r");
      if(fpDB == NULL)
            exit(-1);
      hotelArray = leggiHotels(fpDB, &dim);
      fclose(fpDB);
      hotelArrayFiltrato = filtro(hotelArray, dim, 2, 'M', &dimFiltro);
      for(i = 0; i < dimFiltro; i++)
            printf("%s(%d): %d\n",     hotelArrayFiltrato[i].nome,
                                       hotelArrayFiltrato[i].stelle,
                                       hotelArrayFiltrato[i].postiLiberi);
      free(hotelArray);
      free(hotelArrayFiltrato);
      return 0;
}
```

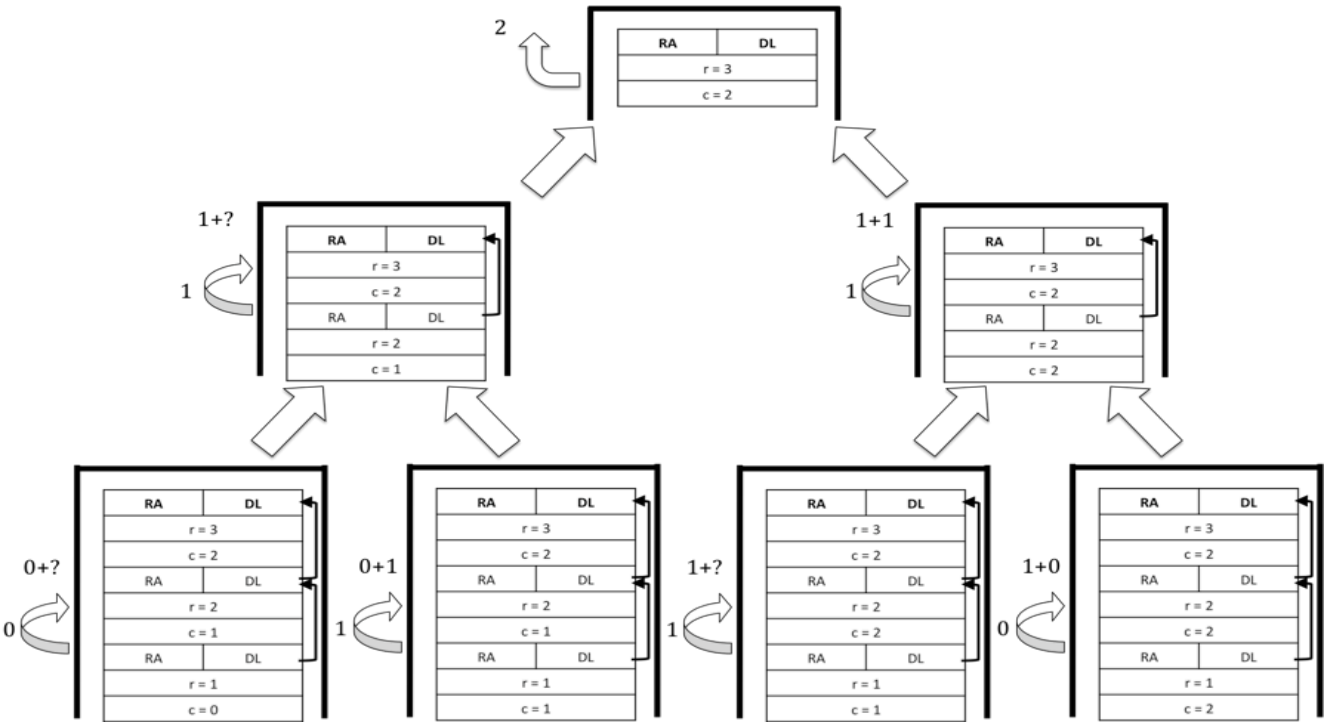
ESERCIZIO 2

```
int somma(list l)
{
      int s = 0;
      while(l != NULL)
      {
            s += l->value;
            l = l->next;
      }
      return s;
}

int sommaPesata(list valori, list pesi)
{
      if(empty(valori))
            return 0;
      else return
            head(valori)*head(pesi) + sommaPesata(tail(valori),tail(pesi));
}


float mediaPesata(list voti, list crediti)
{
      if(lunghezza(voti) == lunghezza(crediti))
            return ((float) sommaPesata(voti,crediti)) / somma(crediti);
      else return -1;
}
```

Il cast a **`float`** è necessario per evitare perdita di informazione nel rapporto tra la somma pesata e il numero totale di crediti.

ESERCIZIO 3

La funzione calcola il valore dell'elemento, all'interno del triangolo di Tartaglia, in riga **`r`** e colonna **`c`**.



ESERCIZIO 5

La stringa **`mn11f3g`** non appartiene alla grammatica data. Il problema si verifica subito all'inizio della frase, con la sequenza **`mn`**. Le lettere **`m`** ed **`n`** possono essere generate solo dal non terminale D; a sua volta, D compare solo nell'espansione del non terminale B. Il problema sorge dal fatto che, stando alla regola di produzione, il non terminale B può essere riscritto solo come DXD. Quindi tra un carattere **`m`** ed un carattere **`n`** deve sempre esserci qualcosa (ottenuto sostituendo il non terminale X).

Si noti inoltre che la grammatica per alcuni frasi può essere ambigua, poiché sia il terminale A che il terminale B possono essere riscritti come X, e lo scopo iniziale può essere riscritto come ABA o BAB.