Fondamenti di Informatica L-A (A.A. 2007/2008) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Giovedì 10 Gennaio 2008 – durata 2h
Compito B

ESERCIZIO 1 (11 punti)

Un sito Internet salva in un file di testo lo stato delle piste di stazioni sciistiche. La prima riga di tale file contiene un intero rappresentante il numero di stazioni registrate nel file mentre ciascuna riga successiva contiene il nome di una stazione sciistica (al più 25 caratteri senza spazi), la quantità minima e massima in centimetri della neve sulle piste di tale stazione (due `float` separati dal carattere `'*'`) e un intero che indica il numero di giorni trascorsi dall'ultimo aggiornamento dello stato della neve (se non specificato diversamente i vari elementi sono separati da spazi). Il candidato, dopo aver definito correttamente una struttura dati chiamata `stazione` in grado di contenere il nome di una stazione sciistica, la quota massima della neve ed i giorni trascorsi dall'ultimo aggiornamento, realizzi:

a) una procedura `void deleteMin(stazione* locs, int* res, int min);`

   che, ricevuti come parametri di ingresso `locs`, un vettore di strutture dati `stazione` già correttamente popolato con `res` elementi, e `min`, un intero, modifichi il vettore `locs` al fine di eliminare quelle località il cui stato della neve sia stato aggiornato da più di `min` giorni. A tale scopo la procedura `deleteMin` deve spostare gli elementi all'interno del vettore `locs` in modo tale che tutti i primi elementi siano stati aggiornati da al più `min` giorni (non è necessario tenere traccia degli elementi che non soddisfano il requisito). Al termine dell'esecuzione della procedura il valore puntato da `res` deve rappresentare il numero dei soli elementi che soddisfano il requisito sul numero minimo di giorni;

b) una funzione `stazione* neve(char* filename, int* res, int giorni);`

   che, ricevuti come parametri di ingresso il nome del file di testo sopra descritto e un intero `giorni`, restituisca un vettore di strutture dati `stazione`, allocato dinamicamente, composto dalle stazioni sciistiche presenti nel file di testo la cui quota minima di neve sia superiore o uguale a 30 centimetri ed il cui stato della neve sia stato aggiornato almeno `giorni` giorni dopo la località aggiornata meno recentemente (la località aggiornata meno recentemente viene individuata a prescindere dalla quota di neve); restituisca inoltre `res`, il numero di elementi presenti nel vettore restituito. A tal fine la funzione `neve()` allochi dinamicamente un vettore che sia in grado di contenere tutte le stazioni contenute nel file di testo e lo popoli con le stazioni la cui quota minima di neve sia almeno 30 centimetri; in seguito invochi opportunamente la procedura al punto a) per eliminare le località che non soddisfano il requisito sul giorno dell'aggiornamento.

ESERCIZIO 2 (9 punti)

Il candidato definisca una funzione *ricorsiva*

```
list extreme(list l, char* first, char* last);
```

che, ricevuti come parametri di ingresso una lista di stringhe ben formate ordinate alfabeticamente e due stringhe ben formate `first` e `last`, restituisca una nuova lista i cui elementi NON siano lessicograficamente (ossia secondo l'ordine alfabetico) contenuti tra `first` e `last` (estremi compresi) e che non contengano lettere maiuscole al loro interno.

Ad esempio, se `l=["alBero", "asino", "bUE", "c2apodanno", "nataLe", "prEsepe", "stel4la"]`, `first="auguri"`, e `last="o"`, la lista restituita sarà `["asino", "stel4la"]` in quanto `"bUE"`, `"c2apodanno"` e `"nataLe"` sono compresi tra `"auguri"` e `"o"` mentre `"alBero"` e `"prEsepe"` contengono caratteri maiuscoli. La funzione deve essere realizzata utilizzando il tipo di dato astratto `list` definito per le stringhe. Si possono utilizzare le operazioni primitive definite durante il corso, che quindi possono NON essere riportate nella soluzione (notare che la funzione `head()` definita per le stringhe restituisce un puntatore al primo carattere della stringa presente nel primo elemento della lista). Si ricorda l'esistenza della funzione di libreria `strcmp(char* str1, char* str2)` che restituisce 0 se le stringhe ben formate `str1` e `str2` sono identiche, un valore inferiore a 0 se `str1` è lessicograficamente inferiore a `str2`, un valore maggiore di 0 altrimenti.

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>
typedef struct l_element{
	char val;
	struct l_element* next;
} l_item;

int a=2;

l_item* update(float* values,char chars[], int dim){
	l_item *last, *new, *res; last=res=NULL;
	for(;a<dim;a++){
		new=(l_item*)malloc(sizeof(l_item));
		new->val=chars[(int)(*(values+a))];
		if(a%2==1){
			new->next=res;
			res=new;
		}
		else{
			new->next=NULL;
			if(last!=NULL) last->next=new;
			else res=new;
			last=new;
		}
	}
	return res;
}

int main(){
	float shift[]={-2.0,+3.2,4.2,0,7,2,8,2.4,-18,11};
	l_item* l;	char str[]="PupazzoDiNeve";
	l=update(shift,str,7);
	while(l!=NULL){
		printf("%c ",l->val);
		l=l->next;
	}
	return 0;
}
```

ESERCIZIO 4 (3 punti)

Si consideri la grammatica `G` con scopo `S`, simboli non terminali `{D, E, F, O, P}` e simboli terminali `{a, b, c, 6, 7, 8, x, y, z}`:

```
S::= PF | PE
O::= ODE | E
P::= OF | DPOD
D::= a | b | c
E::= x | y | z
F::= 6 | 7 | 8
```

La stringa “`azbx6zcy`” appartiene al linguaggio generato da tale grammatica?
In caso affermativo, se ne mostri la derivazione left-most.

ESERCIZIO 5 (3 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):

37 + (-113)

ESERCIZIO 1

```
#define DIM 26
typedef struct{
      char nome[DIM];
      int update;
      float max;
}stazione;

// Soluzione 1
void deleteMin(stazione* locs, int*res, int min){
      int i,j;
      for(i=0;i<*res;i++){
            if(locs[i].update>min){
                  for(j=i;j+1<*res;j++){
                        strcpy(locs[j].nome,locs[j+1].nome);
                        locs[j].update=locs[j+1].update;
                        locs[j].max=locs[j+1].max;
                  }
                  (*res)--;
                  i--;
            }
      }
}

// Soluzione 2
void deleteMin(stazione* locs, int* res, int min){
      int dim=0, i;
      for(i=0; i < *res; i++){
          if((locs + i)->update <= min) {
               *(locs + dim) = *(locs + i);
               dim++;
          }
      }
      *res = dim;
}

stazione* neve(char* filename, int* res, int giorni){
      int righe;
      float min;
      int maxUpdate;
      stazione *resLoc; FILE* file;
      if( (file=fopen(filename,"r"))==NULL ) exit(-1);
      fscanf(file,"%d",&righe);
      resLoc=(stazione*)malloc(sizeof(stazione)*righe);
      maxUpdate=0; *res=0;
      while( (fscanf(file,"%s %f*%f %d", resLoc[*res].nome, &min,
                              &(resLoc[*res].max), &(resLoc[*res].update))) >=0){
            if(resLoc[*res].update>maxUpdate){
                  maxUpdate=resLoc[*res].update;
            }
            if(min>=30.0){
                  (*res)++;
            }
      }
      onlyMin(resLoc,res,maxUpdate-giorni);
      fclose(file);

      return resLoc;
}
```

ESERCIZIO 2

```
list extreme(list l, char* first,char* last){
      char* temp;
      int ok;
      if(empty(l)) return emptylist();
      else if( strcmp(head(l),first)>=0 &&
                  strcmp(head(l),last)<=0 ) return extreme(tail(l),first,last);
      else {
            temp=head(l); ok=1;
            while(*temp!='\0' && ok){
                  if(*temp>='A' && *temp<='Z') ok=0;
                  temp++;
            }
            if(ok) return cons(head(l),extreme(tail(l),first,last));
            else return extreme(tail(l),first,last);
      }
}
```

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
`p P z D i`

Nella funzione `main()` vengono inizializzate le variabili `shift` e `str` e poi viene invocata la funzione `update()` che restituisce una lista di char. Tale funzione itera lungo il vettore `sub` e ad ogni iterazione aggiunge un nuovo elemento alla lista restituita. Il valore del nuovo elemento corrisponde al valore della cella del vettore `chars` il cui indice è uguale al valore dell'a-esima cella del vettore `values`. Il nuovo elemento viene alternativamente aggiunto in coda o in testa alla lista.

ESERCIZIO 4

```
S -> PE -> DPODE -> aPODE -> aOFODE -> aODEFODE -> aEDEFODE -> azDEFODE ->
azbEFODE -> azbxFODE -> azbx6ODE -> azbx6EDE -> azbx6zDE -> azbx6zcE -> azbx6zcy
```

ESERCIZIO 5

```
37  -> 00100101                    00100101 + (37)
113 -> 01110001                    10001111 = (-113)
       10001110                    ----------
-123-> 10001111                    10110100 -> -76
                                   01001011
                                   01001100 ->  76
```