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 A

ESERCIZIO 1 (11 punti)

Un sito Internet salva in un file di testo lo stato delle piste di località sciistiche. La prima riga di tale file contiene un intero rappresentante il numero di località registrate nel file mentre ciascuna riga successiva contiene il nome di una località sciistica (al più 30 caratteri senza spazi), la quantità minima e massima in centimetri della neve sulle piste di tale località (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 `localita` in grado di contenere il nome di una località sciistica e la quota minima e massima della neve, realizzi:

a) una procedura `void onlyMax(localita* locs, int* res, float max);`

   che, ricevuti come parametri di ingresso `locs`, un vettore di strutture dati `localita` già correttamente popolato con `res` elementi, e `max`, un float, modifichi il vettore `locs` al fine di eliminare quelle località la cui quantità massima di neve sia inferiore a `max`. A tale scopo la procedura `onlyMax` deve spostare gli elementi all'interno del vettore `locs` in modo tale che tutti i primi elementi abbiano una quota di neve massima uguale o superiore a `max` (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 sulla quota massima;

b) una funzione `localita* meteo(char* filename, int maxGiorni, int* res);`

   che, ricevuti come parametri di ingresso il nome del file di testo sopra descritto e un intero `maxGiorni`, restituisca un vettore di strutture dati `localita`, allocato dinamicamente, composto dalle località sciistiche presenti nel file di testo il cui stato della neve sia stato aggiornato al più `maxGiorni` prima e la cui quota di neve non sia inferiore di più di 30 centimetri rispetto alla località con più neve (considerando la quota massima e non quella minima delle sole località aggiornate recentemente); restituisca inoltre `res`, il numero di elementi presenti nel vettore restituito. A tal fine la funzione `meteo` allochi dinamicamente un vettore che sia in grado di contenere tutte le località contenute nel file di testo e lo popoli con le località aggiornate da al più `maxGiorni`; in seguito invochi opportunamente la procedura al punto a) per eliminare le località che non soddisfano il requisito sulla quantità di neve.

ESERCIZIO 2 (9 punti)

Il candidato definisca una funzione *ricorsiva*

```
list extract(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 siano lessicograficamente (ossia secondo l'ordine alfabetico) contenuti tra `first` e `last` (estremi compresi) e che non contengano cifre al loro interno.

Ad esempio, se `l=["albero", "asino", "bue", "c2apodanno", "natale", "presepe", "stel4la"]`, `first="auguri"`, e `last="o"`, la lista restituita sarà `["bue", "natale"]` in quanto `"albero"` e `"asino"` precedono `"auguri"`, `"presepe"` e `"stel4la"` sono successivi a `"o"` e `"c2apodanno"` contiene la cifra `'2'`.

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{
	int val;
	struct l_element* next;
} l_item;

int i=1;

l_item* substitute(float values[],int *news, int dim){
	l_item *last, *new, *res; last=res=NULL;
	for(;i<dim;i++){
		new=(l_item*)malloc(sizeof(l_item));
		new->val=values[*(news+i)];
		if(i%2==1){
			new->next=res;
			res=new;
		}
		else{
			new->next=NULL;
			if(last==NULL)last=res;
			last->next=new;
			last=new;
		}
	}
	return res;
}
int main(){
	float add[]={-2.0,+3.2,-4.2,0,37,21,8,-2.4,18,11};
	l_item* l; int n[]={1,3,5,2,7,6,3,-4,2,0};
	l=substitute(add,n,6);
	while(l!=NULL){
		printf("%d ",l->val);
		l=l->next;
	}
	return 0;
}
```

ESERCIZIO 4 (3 punti)

Si consideri la grammatica `G` con scopo `S`, simboli non terminali `{A, B, F, G, H}` e simboli terminali `{x, y, z, l, m, n, 2, 3, 4}`:

```
S::= AF | AH
A::= GABG | BF
B::= BGH | H
F::= x | y | z
G::= l | m | n
H::= 2 | 3 | 4
```

La stringa “`l2n4x3m4`” 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):

29 + (-123)

ESERCIZIO 1

```
#define DIM 31
typedef struct{
	char nome[DIM];
	float min;
	float max;
}localita;

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

// Soluzione 2
void onlyMax(localita* locs, int*res, float max){
	int dim=0, i;
	for(i=0;i<*res;i++){
		if(locs[i].max>=max){
			locs[dim]=locs[i];
			dim++;
		}
	}
	*res=dim;
}

localita* meteo(char* filename, int maxGiorni, int* res){
	int righe,update; float max;
	localita *resLoc; FILE* file;

	if( (file=fopen(filename,"r"))==NULL ) exit(-1);
	fscanf(file,"%d",&righe);
	resLoc=(localita*)malloc(sizeof(localita)*righe);

	max=0; *res=0;
	while( (fscanf(file,"%s %f/%f %d", resLoc[*res].nome,
			&(resLoc[*res].min), &(resLoc[*res].max), &update)) >=0){
		if(update<maxGiorni){
			if(resLoc[*res].max>max){
				max=resLoc[*res].max;
			}
			(*res)++;
		}
	}
	fclose(file);
	onlyMax(resLoc,res,max-30);
	return resLoc;
}
```

ESERCIZIO 2

```
list extract(list l, char* first,char* last){
      char* temp;
      int ok;
      if(empty(l)) return emptylist();
      else if(strcmp(head(l),last)>0) return emptylist();
      else if(strcmp(head(l),first)<0) return extract(tail(l),first,last);
      else {
            temp=head(l); ok=1;
            while(*temp!='\0' && ok){
                  if(*temp>='0' && *temp<='9') ok=0;
                  temp++;
            }
            if(ok) return cons(head(l),extract(tail(l),first,last));
            else return extract(tail(l),first,last);
      }
}
```

ESERCIZIO 3

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

```
8 -4 0 21 -2
```

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

ESERCIZIO 4

```
S -> AH -> GABGH -> lABGH -> lBFBGH -> lBGHFBGH -> lHGHFBGH -> l2GHFBGH ->
l2nHFBGH -> l2n4FBGH -> l2n4xBGH -> l2n4xHGH -> l2n4x3GH -> l2n4x3mH -> l2n4x3m4
```

ESERCIZIO 5

```
29  -> 00011101                    00011101 + (29)
123 -> 01111011                    10000101 = (-123)
       10000100                    ----------
-123-> 10000101                    10100010 -> -94
                                   01011101
                                   01011110 ->  94
```