Fondamenti di Informatica L-A - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista
Prova Parziale d'Esame di Giovedì 11 Febbraio 2010 – durata 2h
Compito Modalità Fondamenti di Informatica L-A (a.a. precedenti)
Compito A

ESERCIZIO 1 (10 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 massima in centimetri della neve sulle piste di tale località (un float) e un intero che indica il numero di giorni (separati da spazi) trascorsi dall'ultima nevicata. Si vuole realizzare un programma che fornisca l'elenco delle località dove c'è poca neve ed ha nevicato da più di un certo numero di giorni.

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 della neve, realizzi:

a) una procedura

```
localita* sadLocations(localita* locs, int dim, int * dimRes, float thres);
```

che, ricevuti come parametri di ingresso **`locs`**, un vettore di strutture dati **`localita`** già correttamente popolato con **`dim`** elementi, e **`thres`**, un float rappresentante una soglia, restituisca in uscita un nuovo vettore, allocato di dimensione minima necessaria **`dimRes`**, contenente tutte le località sfortunate che hanno una quantità di neve inferiore al parametro **`thres`**.

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 in cui abbia nevicato da più di maxGiorni e la cui quota di neve sia inferiore ai 30 centimetri. La funzione restituisca inoltre tramite **`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 sole località dove ha nevicato da più di maxGiorni; in seguito invochi opportunamente la procedura al punto a) per selezionare le località che non soddisfano il requisito sulla quantità di neve, e restituisca tale risultato. Si abbia cura di deallocare eventualme menoria non più necessaria.

ESERCIZIO 2 (10 punti)

Si supponga di avere a disposizione, già definita, l'ADT lista per interi. Il candidato definisca una funzione:

```
list filtraRip(list l1);
```

che, ricevuta in ingresso una lista di interi ordinata, restituisca una lista ottenuta eliminando da **`l1`** tutti gli elementi che sono ripetuti. La funzione dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione dovà essere opportunamente specificata dal candidato.

Ad esempio, se **`l1 = [1, 2, 2, 2, 3, 5, 5, 6]`**, la funzione **`filtraRip(...)`** restituirà la lista **`[1,3, 6]`**.

ESERCIZIO 3 (2 punti)

Si consideri la grammatica G con scopo S, simboli non terminali {A, B, C, D, E} e simboli terminali { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0}:

```
S::= ABA | BAB
A::= C | CA
B::= CD | CDBE
C::= 2 | 4 | 6 | 8
D::= 1 | 3 | 5 | 7 | 9
E::= 0
```

La stringa “212467890” appartiene al linguaggio generato da tale grammatica? In caso affermativo, se ne mostri la derivazione left-most.

ESERCIZIO 4 (3 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 i = 0;

char * filter(char * v, char thres){
	int k;
	int i;
	char * res;
	for (k=0; *(v+k)!='\0'; k++);
	res = (char *) malloc(sizeof(char) * (k));
	i = 0;
	for (k=0;v[k]!='\0';k++) {
		if (v[k]<=thres) {
			*(res+i) = *(v+k);
			i++;
		}
		else
			v[k] = v[(i>0)? i: k];
	}
	res[i] = '\0';
	return res;
}

int main(){
	char * p;
	char str[] = "abcdefg";
	do {
		if (str[i])
			i++;
	} while (str[i]!= '\0');
	p = filter(str, str[i/2]);
	printf("%s\n%s\n", str, p);
	free(p);
	return (0);
}
```

ESERCIZIO 5 (1 punto)

Si descriva sinteticamente il tipo di dato astratto "stack" visto a lezione, introducendo le primitive e fornendo un esempio di uso di tale ADT.

Soluzioni

ESERCIZIO 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define DIM 31
typedef struct{
      char nome[DIM];
      float max;
}localita;

localita* sadLocation(localita* locs, int res, int* dimRes, float thres){
      int i,j;
      localita * result;
      *dimRes = 0;
      for(i=0;i<res;i++)
            if(locs[i].max<thres)
                  *dimRes = *dimRes + 1;
      result = (localita*) malloc(sizeof(localita)* *dimRes);
      j=0;
      for (i=0; i<res; i++) {
            if(locs[i].max<thres) {
                  result[j] = locs[i];
                  j++;
            }
      }
      return result; }

localita* meteo(char* filename, int maxGiorni, int* res) {
      int righe,update;
      float max;
      localita *resLoc;
      localita* result;
      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 %d", resLoc[*res].nome,
                  &(resLoc[*res].max), &update)) >=0)
            if(update>maxGiorni)
                  (*res)++;
      fclose(file);
      result = sadLocation(resLoc, *res, res, 200);
      free(resLoc);
      return result;
}


int main() {
      localita * l;
      int dim;
      int i;
```

```
      l = meteo("loc.txt", 6, &dim);
      for (i=0; i< dim; i++)
            printf("%s %f\n", l[i].nome, l[i].max);
      system("PAUSE");
      return 0;
}
```

ESERCIZIO 2

```
list filtraRip(list l1) {
      element temp;
      int ripetuto;
      if (empty(l1))
            return emptylist();
      else {
            ripetuto = 0;
            temp = head(l1);
            l1 = tail(l1);
            while (!empty(l1) && temp == head(l1) ) {
                  l1 = tail(l1);
                  ripetuto = 1;
            }
            if (!ripetuto)
                  return cons(temp, filtraRip(l1) );
            else
                  return filtraRip(l1);
      }
}
```

ESERCIZIO 3

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S -> BAB -> CDAB -> 2DAB -> 21AB -> 21CAB -> 212AB -> 212CB -> 2124B -> 2124CDBE -> 21246DBE -> 212467BE -> 212467CDE ->2124678DE -> 21246789E -> 212467890.

ESERCIZIO 4

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

La funzione `main()` dapprima calcola la lunghezza della stringa `str`, e poi invoca la funzione `filter(…)`. In particolare, i parametri attuali sono la stringa **`str`** e il carattere di **`str`** posizionato all'indice **`i/2`**: siccome **`i`** vale 7, viene passato il carattere all'indice 3, cioè il carattere '**`d`**'.
La funzione dapprima calcola la dmensione della stringa **`v`**, e poi alloca memoria per un altro array di caratteri di pari dimensione. Quindi scorre la stringa **`v`** fino alla fine, e confronta ogni carattere con il carattere passato tramite il parametro **`thres`**. Se il carattere è "minore o uguale" a **`thres`**, allora viene copiato nel vettore **`res`**; altrimenti la funzione sovrascrive il carattere nel vettore **`v`** passato come parametro. Il valore che vi viene inserito dipende dall'indice **`i`**. Siccome **`thres`** è pari, nell'invocazione, al carattere '**`d`**', vengono copiati in **`res`** i primi quatrto caratteri ("**`abcd`**"). I caratteri successivi sono tutti maggiori di **`thres`**, e quindi nel vettore **`v`** vengono sovrascritti: l'indice **`i`** a quel punto dell'esecuzione vale 4, e quindi viene messo il valore **`v[4]`**, cioè il carattere '**`e`**'.
La funzione `main()` infine stampa a video le stringhe **`str`** e **`p`**.