Fondamenti di Informatica T-1 (A.A. 2011/2012) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 9 Febbraio 2012 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Data una lista di stringhe ben formate `listStr` e un char `ch`, si realizzi una funzione ITERATIVA

```
list atLeastOne(list listStr, char ch);
```

che restituisca una nuova lista contenente gli elementi della lista `listStr` che contengono il carattere `ch`.

Ad esempio se `listStr = [ "paola", "carlo", "federico", "giovanna" ]` e `ch = 'l'`, la funzione `atLeastOne()` deve restituire la lista `[ "paola", "carlo" ]`, ovvero i soli valori della lista `listStr` che contengono il carattere 'l'.

A tal fine si implementi e usi opportunamente la funzione

```
int contains(char* parola, char ch);
```

che restituisce 1 se il carattere `ch` è presente nella stringa ben formata `parola`, 0 altrimenti.

La funzione `atLeastOne()` dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione, compresa la funzione `contains()`, dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `atLeastOne()` creata.

Nota: l'ordine degli elementi della lista restituita dalla funzione `atLeastOne()` è ininfluente.

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 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):

ESERCIZIO 3 (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>

#define DIM 7

int* partition(int* number, int dim, int* start2){
      int *temp=number, count=0, j=0;

      *start2 = 0;

      while( count < dim ){
            if( *temp % 2 != 0 )
                  (*start2)++;
            count++;
            temp++;
      }

      temp = (int*) malloc(sizeof(int)**start2);

      count = 0;
      while( count < dim ){
            if( number[count] % 2 != 0 ){
                  temp[j] = number[count];
                  j++;
            }
            count++;
      }

      return temp;
}

int main() {
      int odd;
      int i[] = {1,2,3,4,5,6,7}, j, *res;
      res = partition(i, DIM, &odd);
      for( j=0; j<odd; j++){
            printf("%d ", res[j]);
      }
      return 0;
}
```

ESERCIZIO 4 (1 punti)

Il candidato illustri brevemente la memoria heap e in quale contesto essa venga utilizzata nella gestione a runtime del linguaggio C.

Soluzioni

ESERCIZIO 1

```
int contains(char* str, char ch){
      int res = 0;
      while( res == 0 && *str != '\0' ){
            if( *str == ch ){
                  res = 1;
            }
            str++;
      }
      return res;
}

list atLeastOne(list listStr, char ch){
      list res;
      res = emptylist();
      while( ! empty(listStr) ){
            if( contains(head(listStr),ch) == 1 ){
                  res = cons(head(listStr), res);
            }
            listStr = tail(listStr);
      }
      return res;
}

int main(){
      list l, res;
      char ch = 'f';
      l = cons("afa", cons("bbb", cons("ccc", cons("dddf", cons("eee", cons("fff",
cons("ggg", emptylist()))))));

      res = atLeastOne(l,ch);
      while( ! empty(res) ){
            printf("%s ", head(res));
            res = tail(res);
      }
      printf("\n");

      return 0;
}
```

ESERCIZIO 2

+ 41 – 113 = -72

+ 41	==>	00101001
+113	==>	01110001
		10001110
-113	==>	10001111
-72	==>	10111000 = -128 + 32 + 16 + 8

ESERCIZIO 3

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

```
1357
```

La funzione `main()` inizializza l'array di int `i` e poi invoca la funzione `partition()`.

La funzione `partition()` conta il numero di interi dispari dell'array `number` e poi alloca dinamicamente spazio di memoria sufficiente a contenere tutti gli interi dispari. Alla fine del primo ciclo `while` il valore referenziato da `start2` è equivalente alla quantità di valori dispari presenti nell'array `number`, ovvero 4.

Il secondo ciclo `while` itera nuovamente lungo `number` e copia i valori dispari nell'area di memoria allocata dinamicamente, mentre i valori pari vengono ignorati. Infine la funzione `partition()` restituisce per valore un puntatore alla prima cella dell'area di memoria allocata dinamicamente e per riferimento la quantità di valori dispari presenti nell'array `number`.

La funzione `main()` stampa sullo standard output i caratteri presenti nell'area di memoria allocata, ovvero i soli valori dispari dell'array `i`.