Fondamenti di Informatica T-1 (A.A. 2016/2017) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Venerdì 16 Giugno 2017 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Si scriva una funzione RICORSIVA `fromPool`

```
int fromPool(list x, list p)
```

che date due liste `x` e `p` di caratteri, dia in uscita `1` se tutti i caratteri di `x` sono contenuti in `p`, `0` altrimenti.

A tale scopo si scriva anche una funzione `int member(char x, list p)` chiamata da `fromPool` che restituisce `1` se `x` appartiene alla lista `p` e `0` altrimenti.

Si realizzi una funzione `main()` che crei le liste `x1={'p','i','p','p','o'}`, `x2={'q','u','o'}` e `p={'i','p','q','o'}` ed utilizzi correttamente la funzione `fromPool` per due volte testando `fromPool(x1,p)` e `fromPool(x2,p)`. I risultati attesi sono rispettivamente `1` e `0`.

La funzione dovrà essere implementata utilizzando le primitive dell'ADT lista, includendo `"list.h"`.

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):
– 15 + 6

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 <stdlib.h>
#include <stdio.h>

int distanceSum(int dx, int dy) {
    return dx+dy;
}

int distanceMax(int dx, int dy) {
    return dx > dy ? dx : dy;
}

int distanceSq(int dx, int dy) {
    return dx * dx + dy * dy;
}

int distance(int xa, int ya, int xb, int yb, int distType) {
    int dx = abs(xa-xb);
```

```
    int dy = abs(ya-yb);
    switch (distType) {
        case 0: return distanceSum(dx,dy);
        case 1: return distanceMax(dx,dy);
        case 2: return distanceSq(dx,dy);
        default: return -1;
    }
}

int mostDistant(int x, int y, int xs[], int ys[], int size, int distType) {
    int i = 0, res = 0;
    int resDist = distance(x,y,xs[i],ys[i],distType);
    for (i = 1; i < size; ++i) {
        int iDist = distance(x,y,xs[i],ys[i],distType);
        if (iDist > resDist) {
            res = i;
            resDist = iDist;
        }
    }
    return res;
}

int main() {
    int x = 1, y = 2;
    int xs[] = {3,-2,1}, ys[] = {0,2,-1};
    printf("%d\n",mostDistant(x,y,xs,ys,3,0));
    printf("%d\n",mostDistant(x,y,xs,ys,3,1));
    printf("%d\n",mostDistant(x,y,xs,ys,3,2));
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si discutano le principali similitudini e differenze tra le strutture dati pila e coda.

Soluzioni

ESERCIZIO 1

```
#include <stdio.h>
#include "list.h"

int member(char x, list p) {
    if (empty(p)) {
        return 0;
    } else {
        if  (head(p) == x) return 1;
        else return member(x,tail(p));
    }
}

int fromPool(list x, list p) {
    if (empty(x)) {
        return 1;
    } else {
        if (member(head(x),p)== 0) return 0;
        else return fromPool(tail(x),p);
    }
}

int main() {
    list p = emptyList();
    list x1 = emptyList();
    list x2 = emptyList();
    p = cons('o',p);
    p = cons('q',p);
    p = cons('p',p);
    p = cons('i',p);
    x1 = cons('o',x1);
    x1 = cons('p',x1);
    x1 = cons('p',x1);
    x1 = cons('i',x1);
    x1 = cons('p',x1);
    x2 = cons('o',x2);
    x2 = cons('u',x2);
    x2 = cons('q',x2);
    printf("%d\n",fromPool(x1,p));
    printf("%d\n",fromPool(x2,p));
    return 0;
}
```

ESERCIZIO 2

– 15 + 6 = – 9
15 = 8 + 4 + 2 + 1 → 00001111
– 15 → 11110001
6 = 4 + 2 → 00000110
– 15 + 6 → 11110111
– 11110111 → 00001001 → 8 + 1 = 9

ESERCIZIO 3

L'output prodotto è

```
0
1
1
```

Le funzioni `distanceSum`, `distanceMax` e `distanceSq` ricevono due parametri interi e ritornano un intero. `distanceSum` ritorna la somma dei due ingressi, `distanceMax` il massimo tra i due ingressi e `distanceSq` la somma dei quadrati dei due ingressi.

La funzione `distance` riceve cinque parametri interi in ingresso e ritorna un intero. Calcola in `dx` e `dy` i valori assoluti delle differenze rispettivamente tra `xa` e `xb` e tra `ya` e `yb`. `dx` e `dy` sono poi usati come parametri nel chiamare, a seconda che il valore di `distType` sia rispettivamente `0`, `1` o `2`, la funzione `distanceSum`, `distanceMax` o `distanceSq`. Il risultato della funzione chiamata è ritornato.

`mostDistant` riceve in ingresso due interi `x` e `y`, due array di interi `xs` e `ys`, e altri due interi `size` e `distType`. Essa per ogni indice `i` inferiore a `size`, chiama `distance(x,y,xs[i],ys[i],distType)`, e ritorna il valore di `i` che porta al più alto valore ritornato da `distance` (il primo tra di essi in caso di pareggio).

Il `main` assegna agli interi `x = 1`, `y = 2`, inizializza due array d'interi `xs = {3,-2,1}` e `ys = {0,2,-1}`, poi stampa il risultato di `mostDistant` chiamato con `size 3`, ossia la lunghezza dei due array `xs` e `ys`, al variare dei tre possibili valori di `distType`.