Fondamenti di Informatica T-1 (A.A. 2012/2013)
Ingegneria Informatica

Prof.ssa Mello

Prova parziale d'esame di Giovedì 12 Settembre 2013 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

Esercizio 1 (6 punti)

Data una lista di stringhe ben formate `lst` e due interi positivi `i` e `n`, si realizzi una funzione ***iterativa***

```
list scarta(list lst, int i, int n);
```

che restituisca una nuova lista contenente gli elementi della lista `lst` a partire da quello in `i`-esima posizione e poi prendendone uno ogni `n`. A tal proposito, l'indice del primo elemento di `lst` sia `1`. Se `lst = ["la", "ape", "vola", "sui", "fiori"]`, `i = 2` e `num = 3`, ad esempio, deve restituire la lista `["ape", "fiori"]`. Se invece `lst` è vuota o `i` è maggiore della dimensione di `lst`, la funzione deve restituire una lista vuota.

La funzione `scarta()` dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione dovrà essere opportunamente specificata dal candidato. L'ordine degli elementi della lista restituita dalla funzione `scarta()` è ininfluente. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `scarta()` appena creata.

Esercizio 2 (2 punti)

Si consideri la seguente funzione:

```
1  int minimax(int h, int i, int[] v) {
2    int t1, t2;
3
4    if (h == 0)
5      return v[i];
6    else {
7      t1 = minimax(h - 1, i * 2 + 0, v);
8      t2 = minimax(h - 1, i * 2 + 1, v);
9      if (h % 2 == 0)
10       return t1 >t2? t1: t2;
11     else
12       return t1 >t2? t2: t1;
13   }
14 }
```

Mostrare la sequenza dei record di attivazione e il valore di ritorno nel caso in cui l'invocazione sia `minimax(1, 1, [1, 3, 3, 7])`.

Esercizio 3 (3 punti)

Si consideri il seguente programma C: compila correttamente? In caso affermativo, cosa viene stampato a video a tempo d'esecuzione? Si motivi opportunamente la risposta.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 10

void funct(char* v, char* r1, char* r2) {
  char* i = v;

  while ((v - i) < SIZE && *v != '\0') {
    if(*v >= 'a' && *v <= 'z') {
      *r1 = *v;
      r1++;
    }
    if(*v >= 'A' && *v <= 'Z') {
      *r2 =  'a' + *v - 'A';
      r2++;
    }
    v += 1;
  }
  *r1 = '\0';
  *r2 = '\0';
  return;
}

int main() {
  char s[SIZE], l[SIZE], u[SIZE];

  strcpy(s, "Bc-Hu:Zk");
  printf("%s\n", s);
  funct(s, l, u);
  printf("%s\n", l);
  printf("%s\n", u);
  return 0;
}
```

Esercizio 4 (1 punto)

Il candidato illustri brevemente la *memoria stack* ed in quale contesto è utilizzata nella gestione a runtime del linguaggio C.

Soluzioni

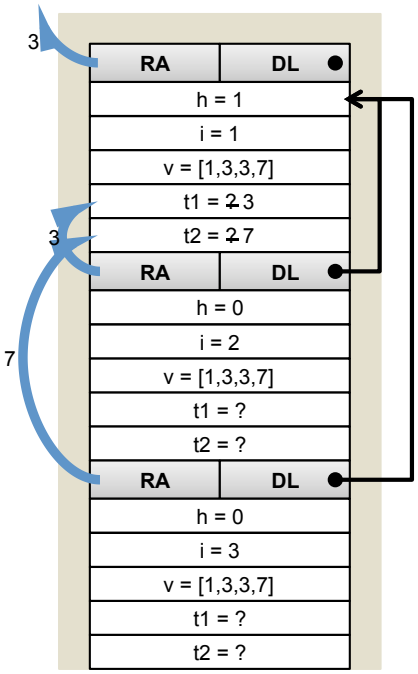
Compito A

Esercizio 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "list.h"

// Funzione iterativa
list scarta(list lst, int i, int n) {
  int k;
  list res = emptylist();

  // Scarta i primi "i" elementi (se esistono)
  for (k = 1; !empty(lst) && k < i; k++) {
    lst = tail(lst);
  }

  // Finche' ci sono elementi
  while (!empty(lst)) {

    // Copia l'elemento attuale
    res = cons(head(lst), res);

    // Scarta i prossimi "n" elementi (se esistono)
    for (k = 1; !empty(lst) && k < n; k++) {
      lst = tail(lst);
    }
  }
  return res;
}

// Funzione main di prova
int main() {
  list l, r;

  printf("scarta\n");
  l = cons("la", cons("ape", cons("vola",
      cons("sui", cons("fiori", emptylist())))));
  r = scarta(l, 2, 3);
  while(!empty(r)) {
    printf("- %s\n", head(r));
    res = tail(res);
  }
  printf("\n");
  return 0;
}
```

Esercizio 2

La funzione restituisce il valore "3".



Esercizio 3

Il programma è sintatticamente corretto e produce la stampa:

```
Bc-Hu:Zk
cuk
bhz
```

La funzione `main()` dichiara tre array di caratteri – `s`, `lc` e `uc` – di dimensione `SIZE` = `10`, copia la stringa ben formata `"Bc-Hu:Zk"` nell'array `s` e lo stampa a video. Quindi invoca la funzione `funct()` con parametri (nell'ordine) `s`, `lc` e `uc`.

La funzione `funct()` copia il puntatore alla stringa `v` in `i`, quindi considera i **`char`** della stringa ben formata `v` uno alla volta fino al terminatore o ad un massimo di `SIZE`. Se il **`char`** in esame è una lettera minuscola, viene accodata alla stringa `r1`. Se invece è una lettera maiuscola, viene convertita in minuscola e accodata alla stringa `r2`. Terminato il ciclo, la funzione accoda un terminatore sia ad `r1` che ad `r2`, in modo da renderle stringhe ben formate. A questo punto `funct()` termina e restituisce `r1` e `r2` per riferimento.

Di nuovo nel `main()`, le stringhe `lc` e `uc` sono stampate a video quindi il programma termina con successo.

Esercizio 4

Vedi slides del corso.