

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

## Ingegneria Informatica

Prof.ssa Mello

Prova parziale d'esame di Giovedì 31 Gennaio 2013 – durata 1h  
Totale 12 punti, sufficienza con 7

### Compito B

#### **Esercizio 1 (6 punti)**

Data una lista di reali `numbers` e due interi `min` e `max` tali che  $0 \leq \text{min} < \text{max}$ , si realizzi una funzione *iterativa*

```
list choose(list numbers, int min, int max);
```

che restituisca una nuova lista contenente gli elementi presenti in `numbers` ad esclusione di quelli compresi tra le posizioni `min` e `max` incluse. L'ordine degli elementi in questa nuova lista è ininfluente. Se `min` o `max` eccedono la dimensione di `numbers`, la funzione considera solo i numeri che riesce a leggere.

Ad esempio, se `numbers = [1.0, 2.7182, 3.1415, 9.8066]`, `min = 2` e `max = 10`, `choose(...)` restituisce la lista `[1.0, 2.7182]`. Se invece `min = 5` e `max = 10`, restituisce la lista di partenza.

Per implementare `choose(...)` si usino le sole primitive dell'ADT `list`. Ogni altra funzione che il candidato desideri utilizzare dovrà essere opportunamente specificata. Si realizzi infine un semplice `main()` di prova che invochi correttamente la funzione `choose(...)` appena creata.

#### **Esercizio 2 (2 punti)**

Si consideri la seguente grammatica  $G = \langle V_T, V_N, P, S \rangle$  con:

- simboli terminali  $V_T = \{ '^', '*', 'x', 'y', '1', '2' \}$
- simboli non-terminali  $V_N = \{ S, A, B, X, Y, Z \}$
- produzioni  $P = \{$ 
  - $S ::= Y \mid Y T \mid Z S$
  - $T ::= Y \mid Y S$
  - $X ::= '^' \mid '*'$
  - $Y ::= 'x' \mid 'y'$
  - $Z ::= '1' \mid '2'$ $\}$
- scopo  $S$

La stringa `'2xyx'` appartiene al linguaggio generato da tale grammatica? In caso affermativo se ne mostri la *derivazione left-most*. Inoltre si disegni il *diagramma sintattico* per la produzione  $S$ .

### Esercizio 3 (3 punti)

Si consideri il seguente programma C: compila correttamente? In caso affermativo, si dica quali valori sono stampati a tempo d'esecuzione motivando opportunamente la risposta.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define SEP '#'
6
7 char* analyse(char* s, int* l) {
8     int i = 0, j = strlen(s), k = j;
9     char* res = (char*) malloc(sizeof(char) * k);
10    *l = 0;
11    do {
12        while (i < k && s[i] == SEP)    i++;
13        while (j > 0 && s[j-1] == SEP)    j--;
14        if (j > 0) {
15            res[*l] = (s[i] != s[j-1]) ? s[i] : '-';
16            i++; j--; (*l)++;
17        }
18    } while(i <= j);
19    res[*l] = '\\0';
20    return res;
21 }
22
23 int main() {
24     int i, dim; char* temp;
25     char sentence[] = "###Etna#Gigante";
26     temp = analyse(sentence, &dim);
27     for (i = 0; i < dim; i++)
28         printf("%c", temp[i]);
29     free(temp);
30     return 0;
31 }
```

### Esercizio 4 (1 punto)

Si descrivano brevemente le differenze tra le invocazioni di funzioni con *passaggio di parametri per valore* e *per riferimento*, avendo cura di fornire un breve esempio in cui tali tecniche sono messe a confronto.

# Soluzioni

## Compito B

### Esercizio 1

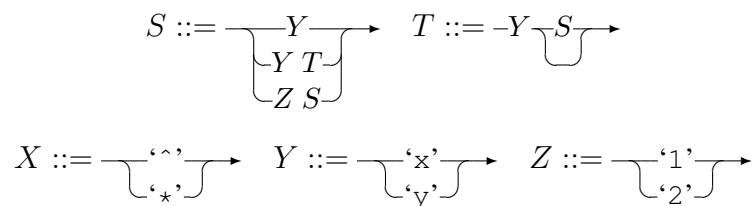
```
1 // Funzione iterativa
2 list choose(list numbers, int min, int max) {
3     int i;
4     list result = emptylist();
5     for (i = 0; !empty(numbers); i++) {
6         if (i < min || i > max)
7             result = cons(head(numbers), result);
8         numbers = tail(numbers);
9     }
10    return result;
11 }
12
13 // Funzione main di prova
14 int main() {
15     list result, numbers = emptylist();
16     numbers = cons(3.1415, cons(9.8066, numbers));
17     numbers = cons(1.0, cons(2.7182, numbers));
18     result = choose(numbers, 2, 10);
19     while (! empty(result)) {
20         printf("%f ", head(res));
21         result = tail(result);
22     }
23     return 0;
24 }
```

### Esercizio 2

La frase '2xyx' appartiene al linguaggio generato dalla grammatica  $G$ . In particolare, è possibile ottenere tale frase mediante la seguente derivazione left-most:

$$S ::= ZS \rightarrow '2' S \rightarrow '2' Y T \rightarrow '2x' T \rightarrow '2x' Y S \rightarrow '2xy' S \rightarrow '2xy' Y \rightarrow '2xyx'$$

Di seguito sono riportati i diagrammi sintattici di tutte le produzioni della grammatica  $G$ :



## Esercizio 3

Il programma è sintatticamente corretto e produce la stampa:

```
E---G-
```

La funzione `main()` introduce le variabili intere `i` e `dim`, le stringhe `temp` e `sentence`, inizializzando quest'ultima a `###Etna#Gigante`. Quindi chiama la funzione `analyse(...)` che accetta due parametri: la stringa `sentence` e `dim` (per riferimento).

La funzione `analyse(...)` introduce a sua volta quattro variabili locali: gli interi `i`, `j` e `k` (rispettivamente inizializzati a 0, `strlen(s)` e `strlen(s)`), e la stringa `res` (per cui viene allocata una quantità di memoria pari a quella occupata da `s`). Si noti che `i` è l'indice per scandire la stringa `s` in avanti, `j` è l'indice per scandire la stringa `s` all'indietro e `k` memorizza la dimensione di `s`. Il parametro `*l` viene impostato a 0.

`analyse(...)` consiste sostanzialmente in un ciclo `do while` che viene ripetuto fintantoché l'indice `i` è minore o uguale a `j`. Al suo interno si trovano due cicli `while` simili tra di loro. Il primo incrementa l'indice `i` fino al primo carattere che non sia il separatore `SEP ('#')` o al termine della stringa, mentre il secondo decrementa `j` fino al primo carattere che non sia `SEP ('#')` o all'inizio della stringa.

Se al termine di queste operazioni gli indici puntano ancora a legittime posizioni di `s`, `analyse(...)` scrive alla posizione `*l` di `res` il carattere `'-'` se i caratteri puntati da `i` e `j` sono diversi tra di loro, o il carattere stesso se invece sono uguali. Quindi l'indice `i` viene incrementato, `j` decrementato e `*l` aggiornato (incrementato).

Al termine del ciclo `do while`, prima di restituire la stringa `res` al chiamante, la funzione `analyse(...)` aggiunge il terminatore di fine stringa `'\0'` per garantire che la stessa stringa `res` sia ben formata.

La funzione `main()` provvede con un apposito ciclo `for` a stampare il contenuto della stringa `res` assegnata alla variabile `temp` un carattere alla volta. Il `main()` quindi libera la memoria allocata per `temp` da `analyse(...)` e termina la propria esecuzione restituendo 0.

## Esercizio 4

Vedi slides del corso.