**Fondamenti di Informatica T-1 (A.A. 2011/2012) - Ingegneria Informatica**
**Prof.ssa Mello**
**Prova Parziale d'Esame di Mercoledì 21 Dicembre 2011 – durata 1h**
**Totale 12 punti, sufficienza con 7**

## Compito B

### ESERCIZIO 1 (6 punti)

Data una lista di caratteri `listaChar` e un intero `dist`, si realizzi una funzione RICORSIVA

```
list atMostDist(list listaChar, int dist);
```

che restituisca una nuova lista contenente gli elementi di `listaChar` che distano al più `dist` posizioni (dal punto di vista lessicografico) rispetto all'elemento successivo.

Ad esempio se `listaChar = ['b', 'g', 'h', 'e', 'a']` e `dist = 3`, la funzione `atMostDist()` deve restituire la lista `['g', 'h']`, ovvero i soli valori della lista `listaChar` che rispettano il requisito dato. Infatti la distanza tra `'g'` e `'h'` è minore o uguale a 3 e la distanza tra `'h'` e `'e'` è minore o uguale a 3.

La funzione `atMostDist()` dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione (ad eccezione della funzione `abs()` della libreria `math`) dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `atMostDist()` creata.

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

### ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione:

```
float ric(float f, int i){
      if( i == 0 ){
            return 0;
      }
      else{
            float res = f / i;
            return res + ric(res, res/i);
      }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (6.6, 2).

**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>
#include <string.h>

char* split(char* str){
	char *temp, *res;

	temp = str;
	while( *temp != '\0' && *temp != ' '){
		temp++;
	}

	res = temp = (char*) malloc(sizeof(char)*(temp-str+1));

	while( *str != '\0' && *str != ' '){

		if( ! ( *str>='A' && *str<='Z' ) ){
			*temp = *str;
		}
		else{
			*temp = '-';
		}
		str++;
		temp++;
	}

	*temp = '\0';

	return res;
}

int main() {
	char* str = "CiaO a23 tuTTi QuAntI";
	char* temp;
	while( *str != '\0' ){
		temp = split(str);
		str = str + strlen(temp) + 1;
		printf("%s\n", temp);
	}
	return 0;
}
```

**ESERCIZIO 4 (1 punti)**

Il candidato illustri brevemente i passaggi necessari per ottenere un‘applicazione eseguibile partendo da più file sorgente.

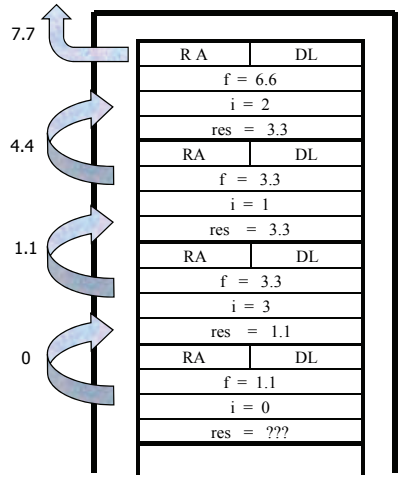
# Soluzioni

## ESERCIZIO 1

```
list atMostDist(list l, int n){
      if( empty(l) || empty(tail(l)) ){
            return emptylist();
      }
      else if( abs( head(l) - head(tail(l)) ) <= n ){
            return cons(head(l), atMostDist(tail(l), n));
      }
      else{
            return atMostDist(tail(l), n);
      }
}


int main(){
      list l, res;
      l = cons('b', cons('g', cons('h', cons('e', cons('a', emptylist())))));

      res = atMostDist(l,3);
      while( ! empty(res) ){
            printf("%c ", head(res));
            res = tail(res);
      }

      return 0;
}
```

## ESERCIZIO 2

La funzione restituisce il valore 7.7.



**ESERCIZIO 3**

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

```
-ia-
a23
tu--i
-u-nt-
```

La funzione `main()` inizializza la stringa ben formata `str` e poi invoca la funzione `split()` finché `str` è diversa dalla stringa vuota; ad ogni iterazione stampa sullo standard output la stringa ben formata restituita dalla funzione `split()` e poi incrementa il valore di `str` del numero di caratteri di cui è composta la stringa ben formata restituita più 1;

La funzione `split()` conta il numero di caratteri della stringa ben formata `str` fino al primo spazio o al terminatore di stringa; poi alloca dinamicamente spazio di memoria sufficiente a contenere tutti i caratteri contati più uno.

Il ciclo `while` itera nuovamente lungo `str` partendo dal primo elemento fino al primo spazio o al terminatore di stringa e lo copia nell'area di memoria allocata dinamicamente; però, se il carattere in esame è di tipo alfabetico maiuscolo, al posto del carattere in esame la funzione `split()` inserisce il carattere '`-`'. Terminato il ciclo `while`, la funzione `split()` inserisce il terminatore di stringa in posizione successiva all'ultimo carattere inserito nell'area di memoria allocata dinamicamente e restituisce al chiamante un puntatore alla prima cella di tale area di memoria.