

**Fondamenti di Informatica T-1 (A.A. 2017/2018) - Ingegneria Informatica**  
**Prof.ssa Mello**  
**Prova Scritta - 9 Febbraio 2018 – durata 1h**  
**Totale 12 punti, sufficienza con 7**

**Compito B**

**ESERCIZIO 1 (6 punti)**

Date due liste `list1` e `list2`, che contengono sempre lo stesso numero di caratteri minuscoli, e un intero positivo `k`, sempre compreso tra 1 e la lunghezza delle due liste, si realizzi una funzione

```
char selectChar(list list1, list list2, int k);
```

che restituisca un carattere `ch`, scelto tra il `k`-esimo elemento di `list1` e il `k`-esimo elemento di `list2` secondo il seguente criterio: se `k` è dispari, va scelto il primo carattere in ordine alfabetico; se `k` è pari, va scelto il secondo dei due (sempre in ordine alfabetico).

Ad esempio se `list1 = ['c', 'a', 't', 't']` e `list2 = ['z', 'f', 'e', 'e']`, la funzione `selectChar` applicata a tali liste deve restituire il carattere `'c'`, se `k = 1` e `'f'` se `k = 2`.

A tal fine si implementi una funzione

```
char trovaEl(int n, list list1);
```

che, dato un intero positivo `n` e una lista `l`, restituisca l'elemento in posizione `n` nella lista in modo ITERATIVO e la si utilizzi opportunamente nella funzione `seleziona`.

Ad esempio la funzione `trovaEl` applicata alla lista `list1` con `n = 2` dovrà restituire `'a'`.

Si supponga per semplicità che `n` sia sempre compreso fra 1 e la lunghezza della lista e che la lista non sia vuota.

Le funzioni dovranno essere implementate utilizzando le sole primitive dell'ADT lista; ogni altra funzione dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `selectChar` sulle seguenti liste: `['g', 'r', 'o', 'e']` e `['f', 'q', 'p', 'c']`, per i valori `k=1` e `k=3`, stampando ogni volta il risultato a terminale (in questo caso `'f'` e `'o'`). Si scrivano le direttive di inclusione necessarie.

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

int func(float* vec, int val){
    float j = 0.5;
    int i = 3;
    while(val <= 0){
        if(vec[--i]*j<2){
            val+=1;
            printf("%d %f\n",val, vec[i]);
        }
        else{
            val=val-1;
        }
    }
    return i;
}

int main() {
    int val=0;
    float fff[]={3.2, 2.5, 5.0, 6.0};
    int res = func(fff, val);
    printf("%d %d", res, val);
}
```

### **ESERCIZIO 4 (1 punto)**

Il candidato illustri brevemente cosa è e a cosa serve la funzione `free()`, realizzando un breve esempio di codice che ne faccia uso correttamente.

# Soluzioni

## ESERCIZIO 1

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

char trovaEl(int n, list l) {
    int i = 1;
    while (i<n) {
        l = tail(l);
        i++;
    }
    return head(l);
}

char selectChar(list list1, list list2, int k) {
    int num;
    char a;
    char b;
    char c;
    a = trovaEl(k, list1);
    b = trovaEl(k, list2);
    if (k%2 != 0){
        c = (a<b) ? a : b
    }
    else{
        c = (a>b) ? a : b
    }
    return c;
}

int main() {
    list list1 = emptyList();
    list list2 = emptyList();
    char ch;
    list1 = cons('e',list1);
    list1 = cons('o',list1);
    list1 = cons('r',list1);
    list1 = cons('g',list1);
    list2 = cons('c',list2);
    list2 = cons('p',list2);
    list2 = cons('q',list2);
    list2 = cons('f',list2);
    ch = selectChar(list1, list2, 1);
    printf("%c", &ch);
    ch = selectChar(list1, list2, 3);
    printf("%c", &ch);

    return 0;
}
```

## ESERCIZIO 2

$5 - 11 = -6$   
 $5 = 4 + 1 = 00000101$   
 $11 = 8 + 2 + 1 = 00001011$   
 $-11 = 11110101$  (rappresentazione in complemento a 2)

5        00000101  
-11     11110101  
5-11    11111010

11111010 (rappresentazione in complemento a 2) = -6

### **ESERCIZIO 3**

L'output prodotto è

```
0 2.5
1 3.2
0 0
```

Il `main` crea un intero `j` inizializzato a `0` e un array di float `fff`. Invoca poi la funzione `func`, passandoli come parametri.

La funzione `func` inizializza un float `j` a `0.5` e un intero `i` a `3`.

Poiché `val` ha valore `0`, si entra nel ciclo `while`.

L'intero `i` viene decrementato, ottenendo `i=2`, poi viene verificato se l'elemento di `vec` in posizione `i`, quindi il float `5.0`, moltiplicato per il float `j` è minore di `2`. Ciò non è vero, perciò si entra nel blocco `else`.

Qui `val` viene decrementato di `1`, ottenendo `val=-1`.

Poiché `val` ha valore negativo, si ricomincia il ciclo `while`.

Di nuovo `i` viene decrementato, ottenendo `i=1`, `vec[i]` è quindi `2.5`, che moltiplicato per `j=0.5` produce un numero minore di `2`, perciò si entra nel blocco `if`.

Il valore `val` viene incrementato di `1`, ottenendo `val=0`, e viene stampato `"0 2.5"`

Poiché `val` ha valore ancora `=0`, si ricomincia il ciclo `while`.

Di nuovo `i` viene decrementato, ottenendo `i=0`, `vec[i]` è quindi `3.2`, che moltiplicato per `j=0.5` produce un numero minore di `2`, perciò si entra nel blocco `if`.

Il valore `val` viene incrementato di `1`, ottenendo `val=1`, e viene stampato `"1 3.2"`

Poiché `val` è maggiore di `0`, si esce dal ciclo `while` e viene restituito `i=0`.

Nel `main` viene quindi assegnato il valore `0` alla variabile `res`, mentre la variabile `val` rimane `=0` poiché era stata passata per valore. L'ultima stampa è quindi `"0 0"`.