**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 A

**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 seleziona(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 `seleziona` 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 elementAt(int n, list list1);
```

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

Ad esempio la funzione `elementAt` 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 `seleziona` sulle seguenti liste: `['g', 'r', 'o', 'e']` e `['f', 'q', 'p', 'c']`, per i valori `k=2` e `k=4`, stampando ogni volta il risultato a terminale (in questo caso `'r'` e `'e'`) . 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):

`9 - 11`

**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 = 0;
    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 `malloc()`, realizzando un breve esempio di codice che ne faccia uso correttamente.

# Soluzioni

## ESERCIZIO 1

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

char elementAt(int n, list l) {
    if (n == 1){
        return head(l);
    }
    else {
        return (subList(n-1, tail(l)));
    }
}

char seleziona(list list1, list list2, int k) {
    int num;
    char a;
    char b;
    char c;
    a = elementAt(k, list1);
    b = elementAt(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 = seleziona(list1, list2, 2);
    printf("%c", &ch);
    ch = seleziona(list1, list2, 4);
    printf("%c", &ch);

    return 0;
}
```

**ESERCIZIO 2**

9 - 11 = - 2
9 = 8 + 1 = 00001001
11 = 8 + 2 + 1 = 00001011
- 11 = 11110101 (rappresentazione in complemento a 2)

| | |
|---|---|
| 9 | 00001001 |
| -11 | 11110101 |
| 9-11 | 11111110 |

11111110 (rappresentazione in complemento a 2) = -2

**ESERCIZIO 3**

L'output prodotto è

```
0 5.0
1 6.0
3 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 `0`.

Poiché `val` ha valore `0`, si entra nel ciclo `while`.
L'intero `i` viene incrementato, ottenendo `i=1`, poi viene verificato se l'elemento di `vec` in posizione `i`, quindi il float `2.5`, moltiplicato per il float `j` è maggiore 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 incrementato, ottenendo `i=2`, `vec[i]` è quindi `5.0`, che moltiplicato per `j=0.5` produce un numero maggiore di `2`, perciò si entra nel blocco `if`.
Il valore `val` viene incrementato di 1, ottenendo `val=0`, e viene stampato "`0 5.0`"

Poiché `val` ha valore ancora `=0`, si ricomincia il ciclo `while`.
Di nuovo `i` viene incrementato, ottenendo `i=3`, `vec[i]` è quindi `6.0`, che moltiplicato per `j=0.5` produce un numero maggiore di `2`, perciò si entra nel blocco `if`.
Il valore `val` viene incrementato di 1, ottenendo `val=1`, e viene stampato "`1 6.0`"

Poiché `val` è maggiore di `0`, si esce dal ciclo `while` e viene restituito `i=3`.
Nel `main` viene quindi assegnato il valore 3 alla variabile `res`, mentre la variabile `val` rimane `=0` poiché era stata passata per valore. L'ultima stampa è quindi "`3 0`".