

**PROVA SCRITTA DEL MODULO DI  
ELEMENTI DI INFORMATICA**  
Corso di Laurea in Ingegneria Biomedica  
2 aprile 2019

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

**ESERCIZIO 1 (4 punti)**

Calcolare il complemento a due del seguente valore binario a cinque bit 11101 e verificare la correttezza del calcolo.

**ESERCIZIO 2 (4 punti)**

1. (2 punti) Spiegare in modo chiaro e sintetico in cosa consiste il modulo ALU di un calcolatore elettronico.
2. (2 punti) Se le parole del modulo di memoria di un calcolatore sono composte da 32 bit, e il numero complessivo di parole è pari a 4096, esprimere la dimensione della memoria in KB.

**ESERCIZIO 3 (25 punti)**

(5 punti) E' dato un file "input.txt" contenente, riga per riga, una coppia di valori di pressione sistolica e diastolica ("massima" e "minima") relativi ad un dato paziente. Si scriva un programma in C che memorizzi su un secondo file "output.txt" la/le coppie di valori letti tale che la differenza fra pressione sistolica e diastolica sia massima.

Per far questo, si definiscano le seguenti strutture dati:

- (1 punto) tipo dati strutturato chiamato `pressione`, costituito da due slot interi `sist` e `diast` che memorizzano appunto una generica coppia di valori di pressione;
- (2 punti) una lista statica chiamata `listaPressione`, il cui elemento di lista è appunto una variabile `p` di tipo `pressione`. Si considerino le funzioni di gestione `CONS`, `HEAD`, `TAIL`, `ISEMPTY` e quella di inizializzazione già implementate.

Si scrivano inoltre le seguenti funzioni:

(3 punti) Funzione `leggiPressione(nomeFile)`: legge da file chiamato `nomeFile` una sequenza di coppie di valori di pressione, le memorizza in una lista di tipo `listaPressione` e la restituisce in uscita.

(6 punti) Funzione `trovaMassimaEscursione(l)`: restituisce un intero corrispondente alla massima differenza sistolica/diastolica rilevata tra i valori presenti nella lista `l`.

(5 punti) Funzione `trovaCoppie(l, esc)`: restituisce una lista di tipo `listaPressione` contenente tutte le coppie della lista `l` la cui differenza sistolica/diastolica è uguale a quella fornita dal parametro intero `esc`.

(3 punti) Funzione `scriviCoppie(l)`: scrive nel file "output.txt" le coppie di valori di pressione sistolica/diastolica presenti nella lista `l`, separandole riga per riga.

**Esempio.** Se il file "input.txt" presentasse le coppie di valori:

```
162 92
180 100
150 80
145 65
```

Il file "output.txt" sarebbe costituito dalle coppie:

```
180 100
145 65
```

**Nota:** l'implementazione corretta delle funzioni di inizializzazione della lista statica, più le funzioni `CONS`, `TAIL`, `HEAD` e `ISEMPTY` darà luogo ad un bonus di 5 punti.

### Soluzione dell'esercizio 1.

Per calcolare il complemento a 2 del valore 11101, prima si invertono i cinque bit ottenendo 00010. Al risultato si somma 1, ottenendo 00011. Per verificare la correttezza del calcolo, basta sommare 11101 e 00011. Si ottiene il valore a sei bit 100000 dove l'1 più significativo è detto "di overflow" e si ignora, mentre i cinque bit meno significativi sono tutti pari a 0 secondo la definizione di complemento a 2.

### Soluzione dell'esercizio 2.

V. dispense del corso.

Poiché  $32 \text{ bit} = 4 \text{ B}$ , e  $4096 = 2^{12}$ , la dimensione della memoria è pari a  $4 * 2^2 * 2^{10} \text{ B} = 16 \text{ KB}$ .

### Soluzione dell'esercizio 3.

```
#include <stdio.h>
#define N 100

/* Definizione dei tipi */

typedef struct{
    int sist, diast;
} pressione;

typedef struct{
    pressione p[N]; /* elementi di lista */
    int n;
} listaPressione;

/*Funzioni di gestione della lista statica: bonus*/

listaPressione inizializza()
{
    listaPressione l;
    l.n=0;
    return l;
}

listaPressione CONS(listaPressione l, pressione p)
{
    l.p[l.n]=p; /*la testa è l'ultimo elemento*/
    l.n++;
    return l;
}

pressione HEAD(listaPressione l)
{
    return l.p[l.n-1]; /*la testa è l'ultimo elemento*/
}

listaPressione TAIL(listaPressione l)
{
    l.n--;
    return l;
}
```

```

int ISEMPY(listaPressione l)
{
    return l.n==0;
}

/* Funzioni dell'esercizio */
listaPressione leggiPressione(char* nomeFile)
{
    FILE *f;
    listaPressione l=inizializza();
    pressione p;

    f=fopen(nomeFile,"r");
    while(!feof(f))
    {
        fscanf(f,"%d %d",&p.sist, &p.diast);
        l=CONS(l,p);
    }

    fclose(f);
    return l;
}

int trovaMassimaEscursione(listaPressione l)
{
    int maxEsc, esc;
    pressione p;

    p=HEAD(l);
    l=TAIL(l);
    maxEsc=p.sist-p.diast;

    while (!ISEMPY(l))
    {
        p=HEAD(l);
        esc=p.sist-p.diast;
        if (esc>maxEsc)
            maxEsc=esc;
        l=TAIL(l);
    }

    return maxEsc;
}

listaPressione trovaCoppie(listaPressione l, int esc)
{
    listaPressione coppie=inizializza();
    pressione p;

    while (!ISEMPY(l))
    {
        p=HEAD(l);
        if (esc==p.sist-p.diast)
            coppie=CONS(coppie,p);
        l=TAIL(l);
    }

    return coppie;
}

```



```

void scriviCoppie(listaPressione l)
{
    FILE *f;
    pressione p;

    f=fopen("output.txt","w");
    while(!ISEMPTY(l))
    {
        p=HEAD(l);
        fprintf(f,"%d %d\n",p.sist,p.diast);
        l=TAIL(l);
    }
    fclose(f);
}

int main()
{
    listaPressione l, c;
    int maxEsc;

    l=leggiPressione("input.txt");
    maxEsc=trovaMassimaEscursione(l);
    c=trovaCoppie(l,maxEsc);
    scriviCoppie(c);

    return 0;
}

```