PROVA SCRITTA DEL MODULO DI
ELEMENTI DI INFORMATICA
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
7 febbraio 2020

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

ESERCIZIO 1 (4 punti)
Indicare il minimo numero di bit necessario per rappresentare il valore 4096 in complemento a 2.

ESERCIZIO 2 (4 punti)
Spiegare in modo chiaro e sintetico in cosa consiste il modulo CPU di un calcolatore elettronico.

ESERCIZIO 3 (25 punti)
(2 punti) Le elezioni sono alle porte e la Sardegna, ottenuta la definitiva indipendenza dall'Italia continentale e (ri)approvata la Carta Delogu in seguito ad un plebiscito, si prepara ad eleggere il suo primo Parlamento. Si presentano alle elezioni solo due partiti, il Partito dei Nuraghe ed il Partito dei Giganti di Mont'e Prama, le cui sigle sono rispettivamente PN e PGM e il sistema elettivo scelto è proporzionale. Lo spoglio delle schede avviene in forma totalmente automatica. I dati di tutti i seggi vengono quindi copiati in un unico file `"voti.txt"` che presenta, per riga, una stringa indicante la sigla del partito prescelto `"PN"` o `"PGM"`, oppure la stringa `"Bianca"` o `"Nulla"` nel caso non sia stata apposta preferenza o la preferenza non sia stata apposta in modo corretto.
Il vostro compito è scrivere un programma C che legga il file suddetto, conti la totalità dei votanti e calcoli la percentuale proporzionale associata a ciascuna delle quattro possibilità. Scriva infine su un file `"risultati.txt"` i risultati ottenuti elencando partiti e casi di schede bianche e nulle in ordine decrescente di percentuale.
Per fare questo, decidete di utilizzare una lista concatenata di variabili del seguente tipo:

```
typedef struct voto
{
    char *preferenza;
    struct voto* successivo;
} listaVoti;
```

Sarà vostro compito tutte le altre eventuali strutture che riterrete utili per scrivere la soluzione.

Scrivete inoltre le seguenti funzioni:

- `(3 punti) leggiVoti(nomefile)`: legge da file nomefile un elenco di stringhe e le memorizza in una variabile di tipo `listaVoti` restituita in uscita.
- `(5 punti) contaPreferenze(voti)`: conta il numero di occorrenze delle stringhe presenti nella variabile `voti`, lista concatenata di tipo `listaVoti`, e restituisce un vettore di interi di 5 componenti, così fatto: la componente 0 contiene il numero complessivo di preferenze presenti in voti, le 1-2 il numero di preferenze `"PN"` e `"PGM"`, le 3-4 il numero di prefereze `"Bianca"` e `"Nulla"`.
- `(9 punti) ordinaVettore(v,n)`: riceve in ingresso un vettore di interi `v` di dimensione `n` e restituisce un altro vettore, della stessa dimensione di `v`, contente i valori degli indici delle componenti di `v` tali che, letti in sequenza da 0 a `n-1`, permettono di leggere i valori di `v` in ordine decrescente. In altre parole se il vettore di indici si chiama `i`, i valori `v[i[0]], …, v[i[n-1]]` sono in ordine decrescente.
- `(6 punti) stampaRisultati(preferenze,indici)`: riceve ingresso due vettori di interi, di 5 componenti, `preferenze` e `indici`, e stampa a video l'elenco delle stringhe associate a ciascun valore di `indici` seguite dal valore corrispondente nel vettore `preferenze`. La stringa corrispondente a `preferenze[0]` equivalente alla somma delle componenti dello stesso `preferenze` da 1 a 4, è `"Totale votanti"`, gli altri indici sono associati ai valori secondo quanto illustrato in relazione alla funzione `contaPreferenze`.

Note
1) La gestione della lista concatenata avviene attraverso le funzioni CONS, TAIL, HEAD e ISEMPTY che si suppongono già implementate. L'implementazione di queste funzioni e l'incremento della modularità del codice con altre funzioni non richieste ma integrate con esse determina un bonus di 5 punti.
2) Non è consentito cambiare i requisiti I/O delle funzioni assegnate nel testo.

Soluzione dell'esercizio 1.

Poiché 4096=2^{12}, per rappresentare tale valore in complemento a 2 occorrono almeno 14 bit. Infatti, i valori positivi in complemento a 2 sono rappresentati con 0 come bit più significativo, mentre i restanti sono rappresentati nella loro forma nota priva di segno. Poiché 4096 presenta un solo 1 nella posizione 12 e i restanti zeri, per rappresentarlo in forma priva di segno occorrerebbero 13 bit[1]. Per completare la rappresentazione in complemento a 2, basta aggiungere 0 in posizione 13, in quanto la commutazione di questo bit a 1 indica, in complemento a 2, una commutazione di segno[2]. Quindi, il numero complessivo di bit è 14.
Un'altra modalità di giungere alla soluzione è osservare, per esempio, che con **3** bit, il massimo intero positivo rappresentabile è 3, pari a $2^2-1=2^{3-1}-1$. Generalizzando, con N bit, il massimo intero positivo rappresentabile è $2^{N-1}-1$. Con 13 bit, si avrebbe un massimo pari a $2^{13-1}-1=2^{12}-1<2^{12}$. Con 14 bit, otteniamo $2^{14-1}-1=2^{13}-1>2^{12}$. Quindi, 14 è il numero minimo di bit richiesto.

Soluzione dell'esercizio 2.

V. dispense del corso.

Soluzione dell'esercizio 3.

```
#include <stdio.h>
#include <stdlib.h>
#define N 5

typedef struct voto
{
    char *preferenza;
    struct voto* successivo;
} listaVoti;

/* Funzioni di gestione della lista concatenata */

listaVoti* CONS(listaVoti* l, char* preferenza)
{
    listaVoti* n;

    n=(listaVoti*)malloc(sizeof(listaVoti));
    n->preferenza=(char*)malloc(sizeof(char)*10);
    strcpy(n->preferenza,preferenza);
    n->successivo=l;

    return n;
}

char* HEAD(listaVoti *l)
{
    return l->preferenza;
}

listaVoti* TAIL(listaVoti *l)
{
    return l->successivo;
}
```

[1] Si ricordi il bit in posizione 0.
[2] Ma non l'inversione del numero! Es. $0111 = +7$ in complemento a 2, ma $1111 \neq -7$.

```
int ISEMPTY(listaVoti *l)
{
    return l==NULL;
}

listaVoti* TAILD(listaVoti *l) /* TAIL con deallocazione */
{
    listaVoti *t;
    t=l->successivo;
    free(l->preferenza);
    free(l);

    return t;
}

/* Funzioni richieste dall'esercizio */


/*leggiVoti(nomefile): legge da file nomefile un elenco di stringhe e le
memorizza in una variabile di tipo listaVoti restituita in uscita.*/

listaVoti* leggiVoti(char* nomefile)
{
    listaVoti *l=NULL;
    FILE *f;
    char p[10];

    f=fopen(nomefile,"r");
    while(!feof(f))
    {
        fscanf(f,"%s",&p[0]);
        l=CONS(l,p);
    }
    fclose(f);
    return l;
}

/*contaPreferenze(voti): conta il numero di occorrenze delle stringhe presenti
nella variabile voti, lista concatenata di tipo listaVoti, e restituisce un
vettore di interi di 5 componenti, così fatto: la componente 0 contiene il
numero complessivo di preferenze presenti in voti, le 1-2 il numero di
preferenze "PN" e "PGM", le 3-4 il numero di preferenze "Bianca" e "Nulla".*/

int* inizializzaVettore(int n)
{
    int i;
    int *v;
    v=(int*)malloc(sizeof(int)*n);

    for (i=0; i<n; i++)
        v[i]=0;

    return v;
}
```

```c
int daiIndice(char* p)
{
    if (!strcmp(p,"PN"))
        return 1;
    if (!strcmp(p,"PGM"))
        return 2;
    if (!strcmp(p,"Bianca"))
        return 3;
    return 4;
}

int* contaPreferenze(listaVoti* voti)
{
    int* numpreferenze;
    int i;
    char* p;

    numpreferenze=inizializzaVettore(N);
    while(!ISEMPTY(voti))
    {
        p=HEAD(voti);
        i=daiIndice(p);
        numpreferenze[i]++;
        numpreferenze[0]++;
        voti=TAIL(voti);
    }

    return numpreferenze;
}

/*ordinaVettore(v,n): riceve in ingresso un vettore di interi v di dimensione n
e restituisce un altro vettore, della stessa dimensione di v, contente i valori
degli indici delle componenti di v tali che, letti in sequenza da 0 a n-1,
permettono di leggere i valori di v in ordine decrescente. In altre parole se il
vettore di indici si chiama i, i valori v[i[0]], …, v[i[n-1]] sono in ordine
decrescente.*/

int* copiaVettore(int* v, int n)
{
    int i;
    int *c;
    c=(int*)malloc(sizeof(int)*n);

    for (i=0; i<n; i++)
        c[i]=v[i];

    return c;
}

void scambia(int *v, int i, int j)
{
    int t;
    t=v[i];
    v[i]=v[j];
    v[j]=t;
}
```

```
int trovaMax(int *v, int start, int stop)
{
    int i, maxv, maxi;

    maxv=v[start];
    maxi=start;
    for(i=start+1; i<stop; i++)
        if(v[i]>maxv)
        {
            maxv=v[i];
            maxi=i;
        }

    return maxi;
}

int* ordinaVettore(int* v, int n)
{
    int i, maxi;
    int *indiciOrdinati, *copiaV;

    indiciOrdinati=inizializzaVettore(n);
    copiaV=copiaVettore(v,n);

    for(i=1; i<n; i++) /*il val in posizione 0 non mi interessa*/
    {
        maxi=trovaMax(v,i,n);
        scambia(copiaV,i,maxi);
        indiciOrdinati[i]=maxi;
    }

    free(copiaV);
    return indiciOrdinati;
}

/*stampaRisultati(preferenze,indici): riceve ingresso due vettori di interi,
di 5 componenti, preferenze e indici, e stampa a video l'elenco delle stringhe
associate a ciascun valore di indici seguite dal valore corrispondente nel
vettore preferenze. La stringa corrispondente a preferenze[0] equivalente alla
somma delle componenti dello stesso preferenze da 1 a 4, è "Totale votanti", gli
altri indici sono associati ai valori secondo quanto illustrato in relazione
alla funzione contaPreferenze.*/

char* codificaIndice(int i)
{
    switch(i)
    {
        case 1: return "Parito dei Nuraghi";
        case 2: return "Partito dei Giganti di Mont\'e Prama";
        case 3: return "Schede bianche";
    }
    return "Schede nulle";
}
```

```
void stampaRisultati(int* preferenze, int* indici)
{
    int i,j,tot;
    float p;
    char* s;

    tot=preferenze[0];

    printf("%35s %5s %6s\n","Partito","Voti","Perc.");
    for(i=1; i<N; i++)
    {
        j=indici[i];
        s=codificaIndice(j);
        p=100.*(float)preferenze[j]/tot;
        printf("%35s %5d %6.2f\%\n",s,preferenze[j],p);
    }
    printf("%35s %5d\n","Totale votanti",tot);

}

void deallocaLista(listaVoti** l)
{
    while(!ISEMPTY(*l))
        *l=TAILD(*l);

    free(*l);
}

int main()
{
    listaVoti* v;
    int *preferenze, *indici;

    v=leggiVoti("200207_voti.txt");
    preferenze=contaPreferenze(v);
    indici=ordinaVettore(preferenze,N);
    stampaRisultati(preferenze, indici);

    free(preferenze);
    free(indici);
    deallocaLista(&v);

    return 0;
}
```