SECONDA PROVA INTERMEDIA DEL MODULO DI
ELEMENTI DI INFORMATICA
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
13 gennaio 2020

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

(10 punti) Un paziente è sottoposto alla lettura del battito cardiaco mediante un opportuno dispositivo che legge il relativo valore in determinati istanti e lo memorizza in un file `"ECG.txt"`. Il dispositivo viene tenuto acceso per un certo tempo.
Alla fine della lettura viene calcolato se i primi cinque valori di pulsazione più elevati risultano strettamente maggiori del 50% rispetto al valor medio lungo tutto il tempo di lettura. Se avviene ciò, è necessario allertare il personale medico con il messaggio `"ALLARME"`.
Scrivere un programma C che, a partire dall'apertura del file dato, effettui i calcoli necessari e stampi a video il messaggio di allarme se necessario.
Si supponga di memorizzare i valori letti da file nella seguente struttura dati:

```
typedef struct
{
   int d[500];
   int n;
} Lista;
```

Le funzioni `CONS, HEAD, TAIL, ISEMPTY` per l'accesso alla lista statica di cui sopra sono già state implementate.
Scrivere inoltre le seguenti funzioni:
1. (5 punti) `leggiDati(nomefile)`: legge da file di nome `nomefile` la sequenza di valori interi e li inserisce in una variabile di tipo `Lista` restituita in uscita.
2. (8 punti) `ordinaVettore(v, n)`: ordina i valori del vettore di interi `v` di dimensione `n` in ordine decrescente a partire dall'indice 0.
3. (10 punti) `calcolaVolte(l)`: restituisce quanti, tra i primi 5 valori massimi in `l`, risultano superiori del 50 % rispetto alla media di tutti i valori di `l`.

Nota: modularizzare ulteriormente il codice scrivendo funzioni non previste dal testo porta ad un bonus massimo di 5 punti.

Soluzione

In questa soluzione è stato inserito codice non necessario, colorato in verde, d'aiuto per studiare il programma.

```
//Soluzione della seconda prova intermedia
#include <stdio.h>
#define N 500

typedef struct
{
   int d[N];
   int n;
} Lista;

//Funzioni CONS, HEAD, TAIL, ISEMPTY
Lista CONS(Lista l, int x)
{
   l.d[l.n]=x;
   l.n++;
   return l;
}

int HEAD(Lista l)
{
   return l.d[l.n-1];
}

Lista TAIL(Lista l)
{
   l.n--;
   return l;
}

int ISEMPTY(Lista l)
{
   return l.n==0;
}

//Funzioni di servizio per modularizzazione
Lista nuovaLista()
{
   Lista nuova;
   nuova.n=0;
   return nuova;
}
```

```c
void scambia (int *v, int i, int j)
{
   int t;

   t=v[i];
   v[i]=v[j];
   v[j]=t;
}

int calcolaMedia(int *v, int n)
{
   int i,m=0;

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

   return m/n;
}

void stampaLista(Lista l)
{
   Lista c=l; //lavoriamo con una copia
   //per non perdere i dati
   int x;

   while(!ISEMPTY(c))
   {
      x=HEAD(c);
      printf("\n%d",x);
      c=TAIL(c);
   }
}

//Funzioni richieste dal problema
Lista leggiDati(char* nomefile)
{
   Lista l;
   FILE *f;
   int x;

   l=nuovaLista(); //anche l.n=0;
   f=fopen(nomefile, "r");
   while (!feof(f))
   {
      fscanf(f,"%d",&x);
      l=CONS(l,x);
   }
   fclose(f);

   return l;
}
```

```
void ordinaVettore(int *v, int n)
//Ordinamento per selezione
{
   int i, j;

   for(i=0; i<n-1; i++)
      for(j=i+1; j<n; j++)
         if(v[j]<v[i])
            scambia(v,i,j);
}

int calcolaVolte(Lista l)
{
   int x, m, n=0;
   Lista c=l; //si lavora con una copia
   //per non perdere l'ordine originario dei valori
   int i;

   printf("\n\nLista iniziale\n");
   stampaLista(l);
   getch();
   ordinaVettore(c.d, c.n);
   printf("\n\nLista ordinata\n");
   stampaLista(c);
   getch();
   m=calcolaMedia(c.d, c.n);
   for(i=0; i<5; i++)
   {
      x=HEAD(c);
      n=n+(x>1.5*m);
      c=TAIL(c);
   }

   return n;
}

int main()
{
   Lista l;
   int n;

   l=leggiDati("ECG.txt");
   printf("\nLista letta");
   n=calcolaVolte(l);
   if(n==5)
      printf("\nALLARME!\n");
   else
      printf("\nSolo %d valori registrati.\n", n);

   return n==5; //restituiamo ad un eventuale
   //processo esterno se c'è allarme
}
```