# mercoledì 8/2
# l compitino

Soluzioni studenti del II turno

**Programmazione)** Scrivere un programma costituito da un main che includa la seguente dichiarazione: int C[100], B[100]. Senza occuparsi di mettere valori dentro C, il main deve riempire B in modo tale che alla fine del programma valga la seguente post-condizione: **POST=(∀a∈[0..99], B[a] >= 0, ∀ valore v ∈C[0..99], esiste un solo b∈[0..99], tale che C[b]=v e B[b]>0 e B[b] è il numero di occorrenze di v in C).**

**Esempio:** C=[5,4,4,4,5,9], B può essere [2,3,0,0,0,1] (o anche[0,0,3,0,2,1] e ci sono anche altri array corretti rispetto alla POST). In ogni caso, B individua i diversi valori contenuti in C (si trovano nelle posizioni 0,1 e 5 di C) e specifica la loro numerosità: B[0]=2 dice che in C ci sono 2 valori C[0]=5, B[1]=3 che ci sono 3 valori C[1]=4 e B[5]=1 che c'è un solo C[5]=9.

La PRE-condizione è vuota. Specificare come commento, **scritto dopo il programma**, invariante e post-condizione per ogni ciclo. Considerando il ciclo più interno (più annidato) del programma, delineare le 3 parti che costituiscono la prova del ciclo.

**Nota**: l'inizializzazione di B è importante.

**PRE=()**

**programma da fare**

**POST=(∀a∈[0..99], B[a] >= 0, ∀ valore v ∈C[0..99], esiste un solo b∈[0..99], tale che C[b]=v e B[b]>0 e B[b] è il numero di occorrenze di v in C).**

L'idea è di scorrere C (indice i) e quando si incontra un valore non ancora "visto", si conta quante volte occorre nel seguito di C, sostituendo questo valore in B[i], si deve anche "neutralizzare" le altre occorrenze in C[j] mettendo B[j]=0. Poiché in B usiamo 0 e un valore >0 per indicare valori già "visti", useremo -1 per indicare quelli non ancora "visti". B va quindi inizializzato a -1.

scorriamo C con un ciclo (indice i) e se B[i]=-1, cioè è visto per la prima volta, allora andiamo con un secondo ciclo (indice j) a cercare le successive occorrenze in C e le neutralizziamo mettendo a 0 B[j] e aumentando B[i] di 1 ogni volta.

Introduciamo una notazione:
POST(k)=(0<=k<100) && (POST con k al posto di 99)=($\forall a \in [0..k]$, B[a] >= 0, $\forall$ valore v $\in$C[0..k], esiste un solo b$\in$[0..k], tale che C[b]=v e B[b]>0 e B[b] è il numero di occorrenze di v in C).

R1=(0<=i<=100) && POST(i-1) && ($\forall a \in [i..99]$, B[a]=-1/0 && B[a]=-1 => C[a] !$\in$C[0..i-1] && B[a]=0=>C[a] $\in$C[0..i-1].

B pieno di -1

```
for(int i=0; i<100; i++) //R1
{
  if(B[i]==-1) //prima volta
   {
     B[i]=1;  //B=B
     for(int j=i+1; j<100;j++) //R2
        if(C[j]==C[i])
            {B[j]=0; B[i]++;}
     //POST2
    }
}//POST1
```

R2=($\forall a \in [i+1..j-1]$ if C[a]=C[i] then B[a]=0 else B[a]=B[a])
&& (B[i] =n. occorrenze di C[i] in C[0..j-1])

POST2=($\forall a \in [i+1..99]$ if C[a]=C[i] then B[a]=0 else
B[a]=B[a]) &&(B[i] =n. occorrenze di C[i] in C[0..99])

```
int C[100],B[100];
for (int i=0;i<100;i++)  //R1
        {
        bool ritrovato=false;
        int d=0;
        for (int j=0;j<100 && ritrovato==false;j++)     //R2
        {
          if (j<i && C[i]==C[j])
                ritrovato=true;
          else
                if (C[i]==C[j])
                        d++;
        }
        //POST2
        B[i]=d;
        }
```

R2=(ritrovato => $\exists a \in [0..j-1]$, a<i, C[a]=C[i], d=0) &&
(!ritrovato => $!\exists a \in [0..j-1]$, a<i, C[a]=C[i], d=n. occ. C[i] in C[0..j-1])

POST2 da R2 con ricetta

```
int C[100], B[100] ;
for (int ai=0; ai<100; ai++) B[ai]=0;

   for (int ci=0; ci<100; ci++) {
   int v=C[ci];
   int indice    =   -1 ; //un valore senz'altro errato
   int occorrenze=    0 ;

    for (int k=0; k<100; k++) {//R2
     if (C[k] == v) { occorrenze++ ; indice=k ; }
   }
   B[indice]=occorrenze;
  }}
```

R2=(occorrenze=n. occ. di  v in C[0..k-1],  indice $\in$[0..k-1] è massimo t.c. C[indice]=v)

```
main(){
  srand (time (NULL));
  int B[100]={};
  int C[100];  int i,j;
  for(i=0;i<100;i++){ //R1=(riempie C[b] di valori casuali)
        C[i]=rand()%100;  }

for(i=0;i<100;i++){//R1
//B=B
    for(j=0;j<100;j++){   //R2
      if(C[i]==C[j]) {
           B[j]++;
      } } }
```

R2=( ogni a $\in$[0..j-1], t.c. C[a]=C[i], B[a]=B[a]+1)

Ma cos'è B?

R1=( per ogni v in C[0..i-1], sia n_v il numero di occ. di v in C[0..i-1], allora per ogni posizione a in [0..99], t.c. C[a]=v, B[a]=n_v)

```
int C[100];   int B[100]={0};
  for(int i=0; i<100; i++) //R1
  {  for(int j=0; j<100; j++) // R2
     {  if(C[i]==C[j])
           B[i]=B[i]+1;
     } //POST2
     for(int z=0; z<100; z++)
     {  if(C[i]==C[z]&&(i!=z))
           B[z]=0;
   }}//POST3
```

POST2=(B[i] = n.  occ.  C[i] in C)

POST3=(per ogni b in [0..99] t.c. b!=i e C[b]=C[i], B[b]=0)