esercizi per il primo compitino

Data la seguente dichiarazione
char X[10][4][10][8];

che tipo e che valore ha l'espressione seguente?
(X+2)[-5]- 4

Si assuma che l'R-valore di X sia X.

Si consideri il seguente programma e si dica se è corretto oppure no spiegando in modo preciso le ragioni della risposta. Si consiglia di usare un grafico che mostri le relazioni tra le diverse variabili.

```
int* F(int x, int & y) {x=y+1; y=x*2; return &y;}

main ()
{int X[3]={1,2,3};  X[1]=*(F(X[0], X[2]))+X[2];
cout<<X[0]<<X[1]<<X[2]<<endl;}
```

Problema: abbiamo array A di dimA interi ordinato e vogliamo eliminare tutte le occorrenze del valore y mantenendo l'array ordinato (con gli elementi diversi da y)

Notazione: (A-y) = quello che resta di A dopo aver eliminato tutte le occ. di y e ricompattato a sinistra gli elementi diversi da y

A=[0, 0, 2, 2, 3, 3, 4] e y=2 A-2=[0,0,3,3,4]

PRE=(A di dimA interi ordinato, y definito, A=vA)
POST=(A=(vA-y), dimA num di elem. di A)

troviamo il primo y in A e quanti y ci sono in A

```
bool trovato=false; int p=0;
for(int i=0; i<dimA && !trovato && !(A[i]>y);i++)
  if(A[i]==y)
      {trovato=true; p=i;}
```

POST=(trovato=>A[p]=y e A[0..p-1]!=y
!trovato=>y non appare in A[0..dimA-1])

R1=(0<=i<=dimA,
A[0..i-2] !=y, trovato=>p=i-1 && A[p]=y
!trovato=> A[0..i-1] !=y)

avendo 3 condizioni nel test di permanenza, la prova del caso di uscita va fatta per bene:

1. trovato (e il resto qualsiasi)
2. !trovato && i=dim*A*
3. !trovato && i<dim*A* && *A*[i]>y

troviamo quanti y ci sono a partire da p

```
int q;
if(trovato)
   for(q=1; p+q<dimA && A[p+q]==y; q++);//R2
```

//POST=(A[p..p+q-1]=y e non si può allungare)

R2=(1<=q<=dimA-p, A[p..p+q-1]=y)

uscita R2 && (p+q=dimA || A[p+q] !=y) cioè non si può allungare

mettiamo tutto in una funzione. input/output ?

```
bool trovato=false; int p=0;
for(int i=0; i<dimA && !trovato && !(A[i]>y);i++)
  if(A[i]==y)
     {trovato=true; p=i;}
int q;
if(trovato)
   for(q=1; p+q<dimA && A[p+q]==y; q++);
```

input: A dimA e y
output: trovato, p e q usiamo una struttura

```
struct tr{bool trovato; int p, q;
tr(bool a=false, int b=0, int c=0){trovato=a;
p=b; q=c;}
};
```

```
tr f1(int*A, int dimA, int y)
{bool trovato=false; int p=0;
for(int i=0; i<dimA && !trovato &&
!(A[i]>y);i++)
  if(A[i]==y)
     {trovato=true; p=i;}
int q;
if(trovato)
   for(q=1; p+q<dimA && A[p+q]==y; q++);

 return tr(trovato,p,q);
}
```

resta il compattamento di A per eliminare gli y
PRE=(A ha dimA elem. 0<=p<p+q<=dimA)

```
int compatta(int*A, int dimA, int p, int q)
{
   int l=dimA-(p+q); //l=elem da spostare
   for(int i=0;i<l; i++) //R3
      A[p+i]=A[p+q+i]

 return dimA-q; //nuovo dimA
}
```

R3=(spostati A[p+q..p+q+i-1] in A[p..p+i-1])
POST=(spostati A[p+q..dimA-1]in A[p..dimA-q-1])

mettiamo insieme le funzioni:

```
main()
{ int A[100], dimA, y;

 leggi( A, dimA, y);
 tr x=f1(A,dimA,y);

if(x.trovato);
  dimA=compatta(A,dimA, x.p, x.q);
....
}
```

soluzione diversa:

```
int resta=0;
for(int i=0; i<dimA; i++)
  if(A[i]!=y)
     if(i > resta)
       {int t=A[resta]; A[resta]=A[i]; A[i]=t;
        resta++;}
     else
        resta++;

 dimA=resta;
```