Prova Intercorso Programmazione I (Tucci/Distasi)

PR1-INT1 MT/RD 10/11/2017

Modello: 1

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Una volta iniziata la prova, non è consentito lasciare l'aula. Usare questi stessi fogli (compreso il retro, dove necessario) per rispondere. *Buon lavoro!*

1. Scrivere una funzione

   ```
   void allargacoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti in cui il primo è minore del secondo, la funzione sottrae `1` al primo ed aggiunge `1` al secondo. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, -8, 10, -100}.
   ```

2. Scrivere una funzione

```
int filtrosum(int a[], int na, int b[], int nb, int c[], int nc);
```

che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno la somma pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

```
b[k] + b[k+1] == a[i].
```

La funzione restituisce il numero di elementi copiati.

Per esempio, se abbiamo

```
a[] = {-7, 9, 33, -100, 21, 6 }
b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
```

dopo l'esecuzione avremo

```
c[] = { 21, 6 }
```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposte per il modello 1

1. Scrivere una funzione

   ```
   void allargacoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti in cui il primo è minore del secondo, la funzione sottrae `1` al primo ed aggiunge `1` al secondo. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, -8, 10, -100}.
   ```

   Risposta: Ecco una possibile soluzione.

   ```
   /*
    * per ogni coppia di elementi adiacenti nell'array a[]
    * in cui il primo elemento e' minore del secondo,
    * sottrai 1 al primo elemento ed aggiungi 1 al secondo.
    */

   void allargacoppie(int a[], int n)
   {
     int i;

     for (i = 0; i < n - 1; i++)   // ci fermiamo al penultimo
       {
         if (a[i] < a[i + 1])
           {
             a[i] -= 1;
             a[i + 1] += 1;
           }
       }
   }
   ```

2. Scrivere una funzione

   ```
   int filtrosum(int a[], int na, int b[], int nb, int c[], int nc);
   ```

   che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

   L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno la somma pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

   ```
   b[k] + b[k+1] == a[i].
   ```

   La funzione restituisce il numero di elementi copiati.

   Per esempio, se abbiamo

   ```
   a[] = {-7, 9, 33, -100, 21, 6 }
   b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
   ```

   dopo l'esecuzione avremo

   ```
   c[] = { 21, 6 }
   ```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposta: Ecco una possibile soluzione.

```
/*
 * test_sum(int x, int filtro[], int size)
 * esiste in filtro[] una coppia adiacente di elementi
 * con somma x? Rispondi NO/SI (0/non 0)
 */

int test_sum(int x, int filtro[], int size)
{
  int i;

  for (i = 0; i < size - 1; i++)        // ci fermiamo al penultimo
    {
      if (filtro[i] + filtro[i + 1] == x)
        {
          return 1;             // si', trovata coppia
        }
    }
  return 0;                     // scandito tutto filtro[], ma coppia non c'e'.
}

/*
 * filtrosum
 * copia elementi da a[] in c[] se superano il filtro in b[]
 * restituisce il numero di elementi copiati
 */
int filtrosum(int a[], int na, int b[], int nb, int c[], int nc)
{
  int i;                        // posizione in a
  int j = 0;                    // posizione in c

  for (i = 0; i < na; i++)
    {
      if (test_sum(a[i], b, nb) != 0)
        {
          c[j++] = a[i];
        }
    }
  return j;
}
```

Prova Intercorso Programmazione I (Tucci/Distasi)

PR1-INT1 MT/RD 10/11/2017

Modello: 2

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Una volta iniziata la prova, non è consentito lasciare l'aula. Usare questi stessi fogli (compreso il retro, dove necessario) per rispondere. *Buon lavoro!*

1. Scrivere una funzione

   ```
   void stringicoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti in cui il secondo è maggiore del primo, la funzione aggiunge `1` al primo e sottrae `2` al secondo. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, -6, 7, -100}.
   ```

2. Scrivere una funzione

```
int filtrodiff(int a[], int na, int b[], int nb, int c[], int nc);
```

che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno la differenza pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

```
b[k] - b[k+1] == a[i].
```

La funzione restituisce il numero di elementi copiati.

Per esempio, se abbiamo

```
a[] = {-7, 9, 33, -100, 21, 6 }
b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
```

dopo l'esecuzione avremo

```
c[] = { 33, 21 }
```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposte per il modello 2

1. Scrivere una funzione

   ```
   void stringicoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti in cui il secondo è maggiore del primo, la funzione aggiunge 1 al primo e sottrae 2 al secondo. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, -6, 7, -100}.
   ```

   Risposta: Ecco una possibile soluzione.

   ```
   /*
    * per ogni coppia di elementi adiacenti nell'array a[]
    * in cui il primo elemento e' minore del secondo,
    * aggiungi 1 al primo elemento e sottrai 2 dal secondo
    */

   void stringicoppie(int a[], int n)
   {
     int i;

     for (i = 0; i < n - 1; i++)   // ci fermiamo al penultimo
       {
         if (a[i] < a[i + 1])
           {
             a[i] += 1;
             a[i + 1] -= 2;
           }
       }
   }
   ```

2. Scrivere una funzione

   ```
   int filtrodiff(int a[], int na, int b[], int nb, int c[], int nc);
   ```

   che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

   L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno la differenza pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

   ```
   b[k] - b[k+1] == a[i].
   ```

   La funzione restituisce il numero di elementi copiati.

   Per esempio, se abbiamo

   ```
   a[] = {-7, 9, 33, -100, 21, 6 }
   b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
   ```

   dopo l'esecuzione avremo

   ```
   c[] = { 33, 21 }
   ```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposta: Ecco una possibile soluzione.

```
/*
 * test_diff(int x, int filtro[], int size)
 * esiste in filtro[] una coppia adiacente di elementi
 * con differenza x? Rispondi NO/SI (0/non 0)
 */

int test_diff(int x, int filtro[], int size)
{
  int i;

  for (i = 0; i < size - 1; i++)        // ci fermiamo al penultimo
    {
      if (filtro[i] - filtro[i + 1] == x)
        {
          return 1;             // si', trovata coppia
        }
    }
  return 0;                     // scandito tutto filtro[], ma coppia non c'e'.
}

/*
 * filtrodiff
 * copia elementi da a[] in c[] se superano il filtro in b[]
 * restituisce il numero di elementi copiati
 */
int filtrodiff(int a[], int na, int b[], int nb, int c[], int nc)
{
  int i;                        // posizione in a
  int j = 0;                    // posizione in c

  for (i = 0; i < na; i++)
    {
      if (test_diff(a[i], b, nb) != 0)
        {
          c[j++] = a[i];
        }
    }
  return j;
}
```

Prova Intercorso Programmazione I (Tucci/Distasi)

PR1-INT1 MT/RD 10/11/2017

Modello: 3

Cognome: _____

Nome: _____

Matricola: _____

Email: _____

Regole del gioco: Compilare i dati personali prima d'incominciare. Una volta iniziata la prova, non è consentito lasciare l'aula. Usare questi stessi fogli (compreso il retro, dove necessario) per rispondere. *Buon lavoro!*

1. Scrivere una funzione

   ```
   void scambiacoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti disposti in ordine crescente, la funzione scambia i due elementi. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, 9, -7, -100}.
   ```

2. Scrivere una funzione

```
int filtroprod(int a[], int na, int b[], int nb, int c[], int nc);
```

che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno il prodotto pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

```
b[k] * b[k+1] == a[i].
```

La funzione restituisce il numero di elementi copiati.

Per esempio, se abbiamo

```
a[] = {-7, 9, 33, -100, 21, 6 }
b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
```

dopo l'esecuzione avremo

```
c[] = { 9, -100 }
```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposte per il modello 3

1. Scrivere una funzione

   ```
   void scambiacoppie(int a[], int n);
   ```

   che prende come parametro un array d'interi `a[]` e lo modifica come segue: per ogni coppia di elementi adiacenti disposti in ordine crescente, la funzione scambia i due elementi. Per esempio, se `a[]` contiene

   ```
   {33, 21, -7, 9, -100},
   ```

   dopo l'esecuzione diventerà

   ```
   {33, 21, 9, -7, -100}.
   ```

   Risposta: Ecco una possibile soluzione.

   ```
   /*
    * per ogni coppia di elementi adiacenti nell'array a[]
    * in cui il primo elemento e' minore del secondo,
    * scambia gli elementi
    */

   void scambiacoppie(int a[], int n)
   {
     int i;
     int tmp;

     for (i = 0; i < n - 1; i++)   // ci fermiamo al penultimo
       {
         if (a[i] < a[i + 1])
           {
             tmp = a[i];
             a[i] = a[i + 1];
             a[i + 1] = tmp;
           }
       }
   }
   ```

2. Scrivere una funzione

   ```
   int filtroprod(int a[], int na, int b[], int nb, int c[], int nc);
   ```

   che ha come parametri 3 array di interi `a[]`, `b[]` e `c[]` con le rispettive lunghezze `na`, `nb`, `nc`.

   L'array `a[]` è la sorgente, `b[]` è il filtro, `c[]` è la destinazione. La funzione esamina l'array `a[]` e copia in `c[]` soltanto gli elementi per cui l'array filtro `b[]` contiene una coppia adiacente di elementi `b[k]` e `b[k+1]` che hanno il prodotto pari all'elemento da copiare. In altri termini, copia `a[i]` se esiste almeno un indice `k` per cui

   ```
   b[k] * b[k+1] == a[i].
   ```

   La funzione restituisce il numero di elementi copiati.

   Per esempio, se abbiamo

   ```
   a[] = {-7, 9, 33, -100, 21, 6 }
   b[] = {55, 34, 1, 3, 3, 20, -5, 26 },
   ```

   dopo l'esecuzione avremo

   ```
   c[] = { 9, -100 }
   ```

e la funzione restituirà 2.

Si può assumere che `nc` $\geq$ `na`, cioè che l'array destinazione sia abbastanza ampio da poterci copiare tutti gli elementi dell'array sorgente.

Suggerimento: scrivere una funzione separata che esamini un elemento di `a[]` e l'array `b[]` per stabilire se l'elemento è da copiare o no.

Risposta: Ecco una possibile soluzione.

```
/*
 * test_prod(int x, int filtro[], int size)
 * esiste in filtro[] una coppia adiacente di elementi
 * con prodotto x? Rispondi NO/SI (0/non 0)
 */

int test_prod(int x, int filtro[], int size)
{
  int i;

  for (i = 0; i < size - 1; i++)        // ci fermiamo al penultimo
    {
      if (filtro[i] * filtro[i + 1] == x)
        {
          return 1;             // si', trovata coppia
        }
    }
  return 0;                     // scandito tutto filtro[], ma coppia non c'e'.
}

/*
 * filtroprod
 * copia elementi da a[] in c[] se superano il filtro in b[]
 * restituisce il numero di elementi copiati
 */
int filtroprod(int a[], int na, int b[], int nb, int c[], int nc)
{
  int i;                        // posizione in a
  int j = 0;                    // posizione in c

  for (i = 0; i < na; i++)
    {
      if (test_prod(a[i], b, nb) != 0)
        {
          c[j++] = a[i];
        }
    }
  return j;
}
```