Corso di Programmazione

II Accertamento del 25 Giugno 2019 / A

cognome e nome

Riporta in modo chiaro negli appositi spazi le soluzioni degli esercizi, oppure precise indicazioni se alcune soluzioni si trovano in un foglio separato. Scrivi inoltre il tuo nome nell'intestazione e su ciascun ulteriore foglio che intendi consegnare.

Selezione degli esercizi proposti.

2. Memoization

Applica la tecnica *top-down* di *memoization* per realizzare una versione più efficiente del seguente programma, che risolve una variante del problema della sottosequenza comune più lunga (calcola la differenza fra due stringhe, restituendo le sequenze di caratteri rimossi rispettivamente da `u` e da `v`):

```
public static String[] diff( String u, String v ) {
  if ( u.equals("") || v.equals("") ) {
    return new String[] { u, v };
  } else if ( u.charAt(0) == v.charAt(0) ) {
    return diff( u.substring(1), v.substring(1) );
  } else {
    String[] x = diff( u.substring(1), v );
    String[] y = diff( u, v.substring(1) );
    if ( x[0].length() < y[0].length() ) {
      return new String[] { u.charAt(0)+x[0], x[1] };
    } else {
      return new String[] { y[0], v.charAt(0)+y[1] };
  }}
}
```

(esercizio 2)

3. Ricorsione e iterazione

Attraverso il metodo statico `st`, il programma ricorsivo seguente calcola i numeri di Stirling del II tipo (problema dei piatti e dei pasticcini). In particolare, è una variante della procedura ricorsiva standard, che utilizza una variabile di stato come contatore. Trattandosi di una variabile condivisa da tutte le invocazioni ricorsive, questa deve essere passata per riferimento, cosa possibile in Java solo con *oggetti*; pertanto il contatore viene rappresentato da un array di un solo elemento. Completa la definizione del metodo `stIter` che trasforma la ricorsione in iterazione applicando uno stack.

```
public static long st( int n, int k ) {  // n, k > 0
  long[] ct = new long[] { 0 };           // contatore: variabile di stato
  sRec( n, k, 1, ct );
  return ct[0];
}

private static void sRec( int n, int k, int q, long[] ct ) {
  if ( (k == 1) || (k == n) ) {
    ct[0] = ct[0] + q;
  } else {
    sRec( n-1, k-1, q, ct );
    sRec( n-1, k, k*q, ct );
  }
}
```

(esercizio 3)

```
public static long stIter( int n, int k ) {  // n, k > 0
  long[] ct = new long[] { 0 };
  Stack<int[]> stack = new Stack<int[]>();

  int[] f = new int[] { ..... };
  stack.push( f );

  while ( ..... ) {

    f = ..... ;

    if ( (f[1] == 1) || ( ..... ) ) {
      ct[0] = ct[0] + f[2];
    } else {

      stack.push( new int[] { f[0]-1, ..... } );

      ..... ;
  }}

  return ..... ;
}
```

4. Verifica formale della correttezza

Dato un intero $n \geq 0$, il seguente metodo statico calcola la parte intera della radice quadrata di n utilizzando solo somme, sottrazioni e confronti. Nel programma sono riportate precondizione, postcondizione e invariante. Introduci opportune espressioni negli spazi previsti in modo tale che i valori assunti dalle variabili soddisfino le relazioni specificate dalle asserzioni. Proponi inoltre una opportuna espressione che definisca i valori della funzione di terminazione.

```
public static int intSqrt( int n ) {   // Pre:  n ≥ 0

  int  q = 0,  x = 0,  y = 2q+1 ,  z = n-y ;

  while ( x <= z ) {                   // Inv:  0 ≤ q ≤ √n ,  x = q^2,  y = 2q+1,  y+z = n

                                       // Term: .....

    q = q + 1;

    x = x + 2q-1 ;
    y = y + 2;

    z = z - 2 ;
  }

  return q ;                           // Post: valore restituito: ⌊√n⌋
}
```

Corso di Programmazione

II Accertamento del 25 Giugno 2019 / B

cognome e nome

Riporta in modo chiaro negli appositi spazi le soluzioni degli esercizi, oppure precise indicazioni se alcune soluzioni si trovano in un foglio separato. Scrivi inoltre il tuo nome nell'intestazione e su ciascun ulteriore foglio che intendi consegnare.

Selezione degli esercizi proposti.

2. Memoization

Applica la tecnica *top-down* di *memoization* per realizzare una versione più efficiente del seguente programma, che risolve una variante del problema della sottosequenza comune più lunga (calcola una misura delle differenze fra due stringhe, restituendo il numero di caratteri rimossi rispettivamente da `u` e da `v`):

```
public static int[] ldiff( String u, String v ) {
  if ( u.equals("") || v.equals("") ) {
    return new int[] { u.length(), v.length() };
  } else if ( u.charAt(0) == v.charAt(0) ) {
    return ldiff( u.substring(1), v.substring(1) );
  } else {
    int[] x = ldiff( u.substring(1), v );
    int[] y = ldiff( u, v.substring(1) );
    if ( x[0] < y[0] ) {
      return new int[] { x[0]+1, x[1] };
    } else {
      return new int[] { y[0], y[1]+1 };
  }}
}
```

(esercizio 2)

3. Ricorsione e iterazione

Attraverso il metodo statico `st`, il programma ricorsivo seguente calcola i numeri di Stirling del II tipo (problema dei piatti e dei pasticcini). In particolare, è una variante della procedura ricorsiva standard, che utilizza una variabile di stato come contatore. Trattandosi di una variabile condivisa da tutte le invocazioni ricorsive, questa deve essere passata per riferimento, cosa possibile in Java solo con *oggetti*; pertanto il contatore viene rappresentato da un array di un solo elemento. Completa la definizione del metodo `stIter` che trasforma la ricorsione in iterazione applicando uno stack.

```
public static long st( int n, int k ) {  // n, k > 0
  long[] cn = new long[] { 0 };           // contatore: variabile di stato
  sRec( 1, n, k, cn );
  return cn[0];
}

private static void sRec( int p, int n, int k, long[] cn ) {
  if ( (k == 1) || (k == n) ) {
    cn[0] = cn[0] + p;
  } else {
    sRec( p, n-1, k-1, cn );
    sRec( k*p, n-1, k, cn );
  }
}
```

(esercizio 3)

```
public static long stIter( int n, int k ) {  // n, k > 0
  long[] cn = new long[] { 0 };
  Stack<int[]> stack = new Stack<int[]>();

  int[] f = new int[] { ..... };
  stack.push( f );

  while ( ..... ) {

    f = ..... ;

    if ( (f[2] == 1) || ( ..... ) ) {
      cn[0] = cn[0] + f[0];
    } else {

      stack.push( new int[] { ..... , f[1]-1, ..... } );

      ..... ;
  }}

  return ..... ;
}
```

4. Verifica formale della correttezza

Dato un intero $n \geq 0$, il seguente metodo statico calcola la parte intera della radice quadrata di n utilizzando solo somme e confronti. Nel programma sono riportate precondizione, postcondizione e invariante. Introduci opportune espressioni negli spazi previsti in modo tale che i valori assunti dalle variabili soddisfino le relazioni specificate dalle asserzioni. Proponi inoltre una opportuna espressione che definisca i valori della funzione di terminazione.

```
public static int intSqrt( int n ) {   // Pre:  n ≥ 0

  int  x = 0,  y = 0,  z = ..... ,  q = ..... ;

  while ( q <= n ) {                   // Inv:  0 ≤ x ≤ √n ,  y = x^2,  z = 2x+1,  q = y+z

                                       // Term: .....

    x = x + 1;
    y = q;

    z = z + ..... ;

    q = q + ..... ;
  }

  return ..... ;   // Post: valore restituito: ⌊√n⌋
}
```