Corso di Programmazione
III Accertamento del 20 Giugno 2005 / A

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro e giustifica sinteticamente le risposte utilizzando lo spazio a disposizione. Dovrai poi consegnare questo testo e le soluzioni, avendo cura di scrivere il tuo nome nell'intestazione e su ciascun foglio.

1. Metodi in Java

Traduci la seguente procedura Scheme in un corrispondente metodo in Java, basato sulla stessa struttura ricorsiva.

```
(define f
  (lambda (x y)  ; x, y naturali
    (cond ((and (= x 0) (= y 0)) 1)
          ((= x 0) (+ (f 0 (quotient y 2)) 1))
          ((= y 0) (+ (f (quotient x 2) 0) 1))
          (else (+ (f (- x 1) (quotient y 2)) (f (quotient x 2) (- y 1))))
          )))
```

2. Memoization

Realizza un programma in Java che applichi la tecnica di memoization alla soluzione ricorsiva dell'esercizio precedente.

3. Asserzioni e invarianti

Modifica il codice del metodo *index*, riportato qui sotto, in modo che la ricerca binaria sia utilizzata per verificare se una delle componenti del vettore *v* ha lo stesso valore di *x* (informalmente: se *x* appartiene a *v*). Assumi che il vettore *v* sia ordinato in ordine *crescente*, ma non contenga necessariamente *x*.

```
public static int index( int x, int[] v ) {
  /** require  v.length > 0;
      ( forall i : { 1 .. v.length-1 } # ( v[i-1] < v[i] ) );
      ( exists j : { 0 .. v.length-1 } # ( v[j] == x ) ); **/
  int l = 0; int r = v.length - 1; int m = -1;
  while ( l < r )
    /** invariant  ( l <= r );  ( exists j : { l .. r } # ( v[j] == x ) ); **/
    /** variant r - l **/  {
    m = (l + r) / 2;
    if ( v[m] < x ) { l = m + 1; }
    else            { r = m;     }
  }
  /** check v[l] == x; **/
  return l;
  /** ensure v[Result] == x; **/
}
```

Apporta quindi le necessarie correzioni alle asserzioni e agli invarianti (puoi esprimere le asserzioni in *Jass* o nel linguaggio matematico, come preferisci). L'intestazione del nuovo metodo deve essere:

```
public static boolean belong( int x, int[] v ) { ... }
```

4. Classi in Java

Definisci in Java una classe *BigNumber* per rappresentare numeri naturali di grandezza potenzialmente arbitraria, cioè non limitati dalle dimensioni delle rappresentazioni a parola fissa come *int* e *long*. Gli oggetti di tipo BigNumber devono essere accessibili attraverso il protocollo specificato qui di seguito:

```
public class BigNumber {
  ...   ...
  public BigNumber() {...}  // Una nuova istanza di BigNumber rappresenta lo zero
  public String rep() {...}  // Restituisce la rappresentazione in base dieci
  public int digit(int i) {...}  // Restituisce l'i-ima cifra decimale del BigNumber
  public void add(BigNumber n) {...}  // Modifica il BigNumber sommando n
}
```

Per esempio, se *x* e *y* sono istanze di *BigNumber*, *x.rep()* e *y.rep()* restituiscono rispettivamente *"124"* e *"51"*, allora dopo aver eseguito l'istruzione *x.add(y)* il valore di *x.rep()* sarà *"175"*, mentre *y.rep()* resterà invariato.

Corso di Programmazione
III Accertamento del 20 Giugno 2005 / B

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro e giustifica sinteticamente le risposte utilizzando lo spazio a disposizione. Dovrai poi consegnare questo testo e le soluzioni, avendo cura di scrivere il tuo nome nell'intestazione e su ciascun foglio.

1. Metodi in Java

Traduci la seguente procedura Scheme in un corrispondente metodo in Java, basato sulla stessa struttura ricorsiva.

```
(define g
  (lambda (x y)  ; x, y naturali
    (cond ((and (< x 2) (< y 2)) 2)
          ((or (< x 2) (< y 2)) (+ (g (quotient (+ x y) 2) 0)
                                   (g 0 (quotient (abs (- x y)) 2))))
          (else (+ (g (quotient (+ x y) 2) (- y 2))
                   (g (- x 2) (quotient (abs (- x y)) 2))))  )))
```

2. Memoization

Realizza un programma in Java che applichi la tecnica di memoization alla soluzione ricorsiva dell'esercizio precedente.

3. Asserzioni e invarianti

Modifica il codice del metodo *index*, riportato qui sotto, in modo che la ricerca binaria sia utilizzata per verificare se una delle componenti del vettore *v* ha lo stesso valore di *x* (informalmente: se *x* appartiene a *v*). Assumi che il vettore *v* sia ordinato in ordine *decrescente*, ma non contenga necessariamente *x*.

```
public static int index( int x, int[] v ) {
  /** require  v.length > 0;
      ( forall i : { 1 .. v.length-1 } # ( v[i-1] > v[i] ) );
      ( exists j : { 0 .. v.length-1 } # ( v[j] == x ) ); **/
  int l = 0; int r = v.length - 1; int m = -1;
  while ( l < r )
    /** invariant  ( l <= r );  ( exists j : { l .. r } # ( v[j] == x ) ); **/
    /** variant r - l **/  {
    m = (l + r) / 2;
    if ( v[m] > x ) { l = m + 1; }
    else            { r = m;     }
  }
  /** check v[l] == x; **/
  return l;
  /** ensure v[Result] == x; **/
}
```

Apporta quindi le necessarie correzioni alle asserzioni e agli invarianti (puoi esprimere le asserzioni in *Jass* o nel linguaggio matematico, come preferisci). L'intestazione del nuovo metodo deve essere:

```
public static boolean belong( int x, int[] v ) { ... }
```

4. Classi in Java

Definisci in Java una classe *BigNumber* per rappresentare numeri naturali di grandezza potenzialmente arbitraria, cioè non limitati dalle dimensioni delle rappresentazioni a parola fissa come *int* e *long*. Gli oggetti di tipo BigNumber devono essere accessibili attraverso il protocollo specificato qui di seguito:

```
public class BigNumber {
  ...   ...
  public BigNumber() {...}  // Una nuova istanza di BigNumber rappresenta lo zero
  public String rep() {...}  // Restituisce la rappresentazione in base due
  public int digit(int i) {...}  // Restituisce l'i-ima cifra binaria del BigNumber
  public void sub(BigNumber n) {...}  // Modifica il BigNumber sottraendo n
}
```

Per esempio, se *x* e *y* sono istanze di *BigNumber*, *x.rep()* e *y.rep()* restituiscono rispettivamente *"1010"* e *"110"*, allora dopo aver eseguito l'istruzione *x.sub(y)* il valore di *x.rep()* sarà *"100"*, mentre *y.rep()* resterà invariato.

Corso di Programmazione

III Accertamento del 20 Giugno 2005 / A

cognome e nome

1. Metodi in Java

```
public static int f(int x, int y) {  // x, y naturali
  if ((x == 0) && (y == 0)) {
    return 1;
  } else if (x == 0) {
    return f(0,y/2) + 1;
  } else if (y == 0) {
    return f(x/2,0) + 1;
  } else {
    return f(x-1,y/2) + f(x/2,y-1);
  }
}
```

2. Memoization

```
public static int f( int x, int y ) {

  int[][] history = new int[x+1][y+1];
  for (int i=0; i<=x; i=i+1) {
    for (int j=0; j<=y; j=j+1) {
      history[i][j] = 0;
  }}
  return f_mem(x,y,history);
}


public static int f_mem( int x, int y, int[][] history ) {

  if ( history[x][y] == 0 ) {
    if ((x == 0) && (y == 0)) {
      history[x][y] = 1;
    } else if (x == 0) {
      history[x][y] = f_mem(0,y/2,history) + 1;
    } else if (y == 0) {
      history[x][y] = f_mem(x/2,0,history) + 1;
    } else {
      history[x][y] = f_mem(x-1,y/2,history) + f_mem(x/2,y-1,history);
  }}
  return history[x][y];
}
```

3. Asserzioni e invarianti

```
public static boolean belong( int x, int[] v ) {
  /**  require  v.length > 0;
         ( forall i : { 1 .. v.length-1 }  #  ( v[i-1] < v[i] ) );
  **/
  int l = 0;  int r = v.length – 1;  int m = -1;
  while ( l < r )
    /**  invariant  ( !(exists j : { 0 .. v.length-1 } #
                                    (v[j] == x)) || ((v[l] <= x) && (x <= v[r])) );
    **/
    /**  variant  r - l  **/  {
    m = (l + r) / 2;
    if ( v[m] < x ) {  l = m + 1;  }
    else            {  r = m;      }
  }
  /**  check  ( !(exists j : { 0 .. v.length-1 } #
                              (v[j] == x)) || (v[l] == x) );
  **/
  return ( v[l] == x );
  /**  ensure  ( Result == (exists j : { 0 .. v.length-1 } # (v[j] == x)) );  **/
}
```

4. Classi in Java

```
public class BigNumber {

  private String big;

  public BigNumber() {  // Una nuova istanza di BigNumber rappresenta lo zero
    big = "0";
  }

  public String rep() {  // Restituisce la rappresentazione in base dieci
    return big;
  }

  public int digit( int i ) {   // Restituisce l'i-ima cifra decimale del BigNumber
    int k = big.length();
    if ( i < k ) {
      return Integer.parseInt( big.substring(k-i-1,k-i) );
    } else {
      return 0;
    }
  }

  public void add( BigNumber n ) {  // Modifica il BigNumber sommando n
    String sum = "";
    int carry = 0;
    for ( int i=0; i<Math.max(big.length(),n.big.length()); i=i+1 ) {
      int d = digit(i) + n.digit(i) + carry;
      sum = "" + ( d % 10 ) + sum;
      carry = d / 10;
    }
    if ( carry > 0 ) {
      sum = "1" + sum;
    }
    big = sum;
  }
}
```

Corso di Programmazione

III Accertamento del 20 Giugno 2005 / B

cognome e nome

1. Metodi in Java

```
public static int g(int x, int y) {  // x, y naturali
  if ((x < 2) && (y < 2)) {
    return 2;
  } else if ((x < 2) || (y < 2)) {
    return g( (x+y)/2, 0 ) + g( 0, Math.abs(x-y)/2 );
  } else {
    return g( (x+y)/2, y-2 ) + g( x-2, Math.abs(x-y)/2 );
  }
}
```

2. Memoization

```
public static int g( int x, int y ) {  // x, y naturali

  int z = Math.max(x,y);
  int[][] history = new int[z+1][z+1];
  for (int i=0; i<=z; i=i+1) {
    for (int j=0; j<=z; j=j+1) {
      history[i][j] = 0;
  }}
  return g_mem(x,y,history);
}


public static int g_mem( int x, int y, int[][] history ) {

  if ( history[x][y] == 0 ) {
    if ((x < 2) && (y < 2)) {
      history[x][y] = 2;
    } else if ((x < 2) || (y < 2)) {
      history[x][y] = g_mem( (x+y)/2, 0, history )
                      + g_mem( 0, Math.abs(x-y)/2, history );
    } else {
      history[x][y] = g_mem( (x+y)/2, y-2, history )
                      + g_mem( x-2, Math.abs(x-y)/2, history );
  }}
  return history[x][y];
}
```

3. Asserzioni e invarianti

```
public static boolean belong( int x, int[] v ) {
  /**  require  v.length > 0;
         ( forall i : { 1 .. v.length-1 }  #  ( v[i-1] > v[i] ) );
   **/
  int l = 0;  int r = v.length – 1;  int m = -1;
  while ( l < r )
    /**  invariant  ( !(exists j : { 0 .. v.length-1 } #
                                    (v[j] == x)) || ((v[l] >= x) && (x >= v[r])) );
    **/
    /**  variant  r - l  **/  {
    m = (l + r) / 2;
    if ( v[m] > x ) {  l = m + 1;  }
    else            {  r = m;      }
  }
  /**  check  ( !(exists j : { 0 .. v.length-1 } # (v[j] == x)) || (v[l] == x) );
  **/
  return ( v[l] == x );
  /**  ensure  ( Result == (exists j : { 0 .. v.length-1 } # (v[j] == x)) );  **/
}
```

4. Classi in Java

```
public class BigNumber {

  private String big;

  public BigNumber() {  // Una nuova istanza di BigNumber rappresenta lo zero
    big = "0";
  }

  public String rep() {  // Restituisce la rappresentazione in base dieci
    return big;
  }

  public int digit( int i ) {   // Restituisce l'i-ima cifra decimale del BigNumber
    int k = big.length();
    if ( i < k ) {
      return Integer.parseInt( big.substring(k-i-1,k-i) );
    } else {
      return 0;
    }
  }

  public void sub( BigNumber n ) {  // Modifica il BigNumber sottraendo n
    String dif = "";
    int borrow = 0;
    for ( int i=0; i<big.length(); i=i+1 ) {
      int d = digit(i) - n.digit(i) - borrow;
      if ( d < 0 ) {  borrow = 1;  d = d + 2;  }
      else         {  borrow = 0;              }
      dif = "" + d + dif;
    }
    int k = 0;
    while ( (k < dif.length() - 1) && (dif.charAt(k) == '0') ) {
      k = k + 1;
    }
    big = dif.substring(k);
  }
}
```