Domanda **1**

Risposta non ancora data

Punteggio max.: 1,00

1. Argomenti e valori procedurali

La procedura *list-freq*, definita qui sotto, calcola il numero complessivo di occorrenze del carattere *c* nelle stringhe contenute in una lista *lst*:

```
(define list-freq  ; val: intero
  (lambda (c lst)  ; c: carattere, lst: lista di stringhe
    (proc (map (str-freq c) lst))
    ))
```

A tal fine, si applica la procedura predefinita *map* per "mappare" la lista di stringhe *lst* in una lista di interi, dove ciascun intero corrisponde al numero di occorrenze di *c* in una delle stringhe di *lst*; quindi *proc* somma gli interi della lista ottenuta.

Per esempio, se *lst* = ("giglio" "rosa" "iris" "viola") e *c* = #\i, i due passaggi possono essere sintetizzati informalmente così:

("giglio" "rosa" "iris" "viola") → (2 0 2 1) → 5

1.1. Definisci la procedura *str-freq*.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

1. Argomenti e valori procedurali

1.2. Definisci la procedura *proc.*

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Feedback specifico: Visualizzato

Feedback generale: Visualizzato

Domanda **1**

Risposta non ancora data

Punteggio max.: 1,00

1. Argomenti e valori procedurali

La procedura *list-freq*, definita qui sotto, calcola il numero più elevato di occorrenze del numero *n* in una lista (numerica) contenuta nella lista *lst*:

```
(define list-freq  ; val: intero
  (lambda (n lst)  ; n: numero, lst: lista di liste numeriche
    (proc (map (inl-freq n) lst))
    ))
```

A tal fine, si applica la procedura predefinita *map* per "mappare" la lista di liste *lst* in una lista di interi, dove ciascun intero corrisponde al numero di occorrenze di *n* in una delle liste di *lst*; quindi *proc* determina il massimo fra gli interi della lista ottenuta.

Per esempio, se *lst* = ((1 2 3) (4 4 4) (2 4 6 4 2) (6 4 2)) e *n* = 4, i due passaggi possono essere sintetizzati informalmente così:

((1 2 3) (4 4 4) (2 4 6 4 2) (6 4 2)) → (0 3 2 1) → 3

1.1. Definisci la procedura *inl-freq*.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

1. Argomenti e valori procedurali

1.2. Definisci la procedura *proc*.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Feedback specifico: Visualizzato

Feedback generale: Visualizzato

Domanda 1

Risposta non ancora data

Non valutata

2. Memoization

Considera il seguente programma basato su una ricorsione ad albero:

```
public static int fun( int[] u, int[] v ) {
  return rec( u, v, 0, 0 );
}
private static int rec( int[] u, int[] v, int i, int j ) {
  int m = u.length;
  int n = v.length;
  if ( (i == m) || (i+j == n) ) {
    return  0;
  } else if ( u[i] == v[i+j] ) {
    return  1 + rec( u, v, i+1, j );
  } else {
    return  Math.max( rec(u,v,i+1,j-1), rec(u,v,i,j+1) );
  }
}
```

2.1. Supponi che nel corso dell'esecuzione di un programma che utilizza *fun* venga valutata l'espressione:

fun(new int[] {1, 2, 3}, new int[] {4, 5, 6, 7})

Questa valutazione si svilupperà attraverso successive invocazioni ricorsive di *rec* per diversi valori degli argomenti. Quali sono il valore più piccolo e il valore più grande che assumeranno i parametri *i* e *j* nelle invocazioni ricorsive di *rec*?

- Valore più piccolo di *i* : [ ] e valore più grande di *i* : [ ] ;
- Valore più piccolo di *j* : [ ] e valore più grande di *j* : [ ] .

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 0

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

2. Memoization

2.2. Applica una tecnica *top-down* (ricorsiva) di memoization per realizzare un programma più efficiente.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo

Domanda **1**

Risposta non ancora data

Non valutata

3. Programmazione orientata agli oggetti in Java

Data la disposizione delle regine in una scacchiera *n* x *n*, rappresentata da una lista di stringhe di tipo *SList<String>* dove ciascuna stringa codifica le coordinate della regina con una lettera (colonna) e una cifra (riga), la procedura *checkSolution* verifica se tale disposizione corrisponde a una soluzione del rompicapo delle *n* regine.

Per esempio, nel caso *n* = 4 la lista ("b1", "d2", "a3", "c4") denota una soluzione, mentre ("b1", "d2", "a3") o ("a1", "c2", "b3", "d4") no.

3.1. Completa il programma impostato qui sotto per verificare se una disposizione di regine denota una soluzione del rompicapo dell *n* regine, dove le procedure *row* e *col* "estraggono" le coordinate intere di riga e colonna, rispettivamente, da una stringa di due caratteri che rispetta le convenzioni consuete della notazione scacchistica. Il programma opera provando ad inserire le regine una dopo l'altra e controllando di volta in volta che non si producano conflitti. La versione della classe *Board* utilizzata è l'ultima discussa in classe, che prevede l'evoluzione dello stato aggiungendo o rimuovendo regine (definizione di Board).

```
public static boolean checkSolution( int n, SList<String> queens ) {
  return checkCompletion( new Board(n), queens );
}
private static boolean checkCompletion( Board b, SList<String> queens ) {
  int n = b.size();
  int q = b.queensOn();
  if (  ) {
    return ( q == n );
  } else {
    int i = row( queens.car() );
    int j = col( queens.car() );
    if (  ) {
      return false;
    } else {
      ;
      return checkCompletion(  );
  }}
}
```

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 0

Ricomincia con queste opzioni

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

3. Programmazione orientata agli oggetti in Java

3.2. Assumendo che la dimensione *n* della scacchiera sia minore o uguale a 9, definisci le procedure (metodi statici) *row* e *col*.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Domanda 1

Risposta non ancora data

Non valutata

3. Programmazione orientata agli oggetti in Java

Data la disposizione delle regine in una scacchiera *n* x *n*, rappresentata da una lista di stringhe di tipo *SList<String>* dove ciascuna stringa codifica le coordinate della regina con una lettera (colonna) e una cifra (riga), la procedura *checkSolution* verifica se tale disposizione corrisponde a una soluzione del rompicapo delle *n* regine.

Per esempio, nel caso *n* = 4 la lista ("b1", "d2", "a3", "c4") denota una soluzione, mentre ("b1", "d2", "a3") o ("a1", "c2", "b3", "d4") no.

3.1. Completa il programma impostato qui sotto per verificare se una disposizione di regine denota una soluzione del rompicapo dell *n* regine, dove le procedure *row* e *col* "estraggono" le coordinate intere di riga e colonna, rispettivamente, da una stringa di due caratteri che rispetta le convenzioni consuete della notazione scacchistica. Il programma opera provando ad inserire le regine una dopo l'altra e controllando di volta in volta che non si producano conflitti. La versione della classe *Board* utilizzata è l'ultima discussa in classe, che prevede l'evoluzione dello stato aggiungendo o rimuovendo regine (definizione di Board).

```
public static boolean checkSolution( int n, SList<String> queens ) {
  Board b = new Board( n );
  int q = 0;
  while (  ) {
    int i = row( queens.car() );
    int j = col( queens.car() );
    if (  ) {
      return false;
    } else {
      ;
      ;
    }
    queens =  ;
  }
  return ( q == n );
}
```

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 0

Ricomincia con queste opzioni

Opzioni di visualizzazione

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

3. Programmazione orientata agli oggetti in Java

3.2. Assumendo che la dimensione *n* della scacchiera sia minore o uguale a 9, definisci le procedure (metodi statici) *row* e *col*.

Soluzione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Domanda 1

Risposta non ancora data

Non valutata

4. Correttezza dei programmi iterativi

La procedura *manh* risolve il problema dei percorsi di Manhattan. In corrispondenza al programma iterativo che realizza *manh* sono riportate precondizione (*Pre*), postcondizione (*Post*) e invariante (*Inv*). In particolare, si intende che l'invariante consente di dimostrare la correttezza del programma in relazione a quanto specificato da precondizione e postcondizione.

4.1. Completa la definizione della procedura *manh* in modo che le proprietà formalizzate nell'invariante siano soddisfatte, cioè in modo che si possa dimostrare la correttezza parziale del programma.

```
public static long manh( int i, int j ) {  // Pre:  i, j ≥ 0
  long m =                                                    ;
  long n =                                                    ;
  int  k = 1;
  while ( k <= i ) {  // Inv:  m = (j+k-1)! / ( j! · (k-1)! )  ∧  n = (j+k)! / ( j! · k! )
    long x = n + ( m * (j + k) *                              ) / ( (k + 1) * k );
    m = n;
    n = x;
    k = k + 1;
  }
  return m;  // Post:  m = (i+j)! / ( i! · j! )
}
```

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 0

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Feedback specifico: Visualizzato

Domanda 1

Risposta non ancora data

Punteggio max.: 1,00

4. Correttezza dei programmi iterativi

4.2. Quale potrebbe essere la funzione di terminazione per completare la dimostrazione di correttezza?

Introduci un'espressione appropriata per determinarne il valore in relazione allo stato della computazione:

Ricomincia | Salva | Inserisci le risposte esatte | Invia e termina | Chiudi anteprima

Informazioni tecniche

Esporta la domanda nel formato Moodle XML

Minimizza tutto

Opzioni per il tentativo

Comportamento della domanda: Feedback differito

Punteggio massimo: 1

Ricomincia con queste opzioni

Opzioni di visualizzazione

Se corretto: Visualizzato

Punteggio: Visualizza punteggio e max.

Cifre decimali nelle valutazioni: 2

Feedback specifico: Visualizzato

Feedback generale: Visualizzato

Risposta corretta: Visualizzato

Storico delle risposte: Non visualizzata

Aggiorna opzioni di visualizzazione