Corso di laurea in Informatica AA 2006/07 Programmazione, M.A. Alberti

III Appello 2006/07 - 14 giugno 2007

Cognome **Nome**

Matricola

1

Assumendo la dichiarazione:

```
Random rand = new Random ();
```

Indicare il range dei valori delle seguenti dichiarazioni:

`rand.nextInt() % 10;` **[-9, 9] estremi inclusi**

`(int) (Math.random () * 15);` **[0, 14] estremi inclusi**

Inoltre scrivere un'istruzione per produrre valori paseudo-casuali nell'intervallo:

`0 – 10` usando l'oggetto `rand` **`Math.abs(rand.nextInt() % 11);`**

`5 – 25` usando il metodo `random ()` della classe `Math` **`(int)(Math.random()*21) + 5;`**

2

Generare un numero `DIM` di numeri pseudo causali interi positivi minori di `MAX` e archiviarli, se sono pari, in un array che verrà riportato all'ambiente chiamante.

```
public static int[] popola () {
  int[] archivio = new int[DIM];
  int numero;
  for (int i=0; i<DIM; i++) {
        do
              numero = (int)(Math.random()* MAX);
        while (numero%2 != 0);
        archivio[i] = numero;
  }
  return archivio;
}
```

3

Data la seguente funzione ricorsiva:

```
public static int f(int m, int n) {
   if (m > 5)
       return n;
   else if (m > n)
            return 1 + f(m+2, n-1);
        else
            return 1 - f(n+1, m-2);
}
```

calcolare il valore riportato dalla chiamata `f(4, 3)`: **3**

calcolare il valore riportato dalla chiamata `f(2, 7)`: **1**

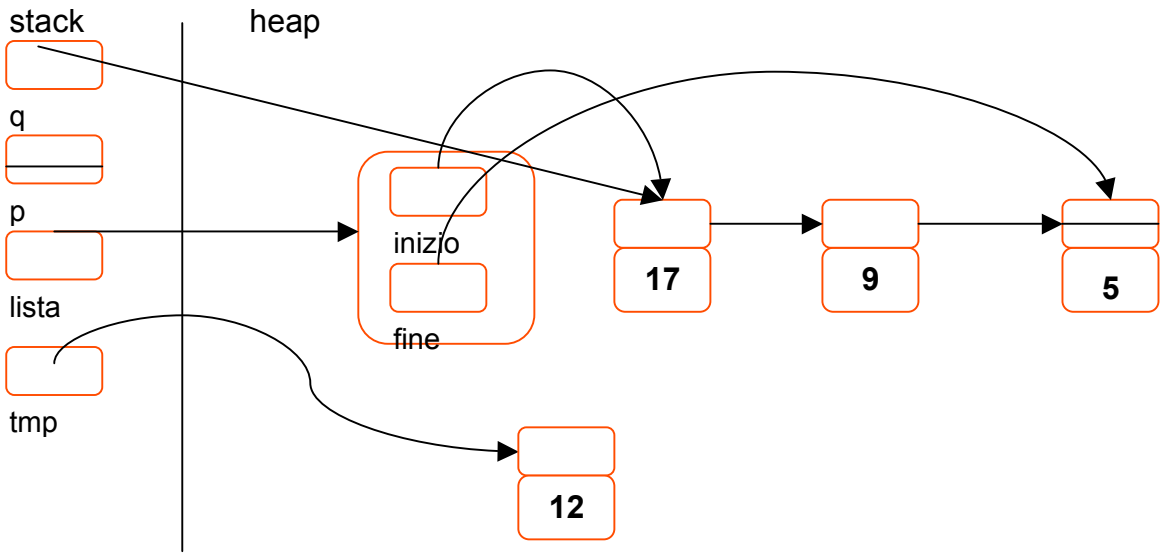
calcolare il valore riportato dalla chiamata `f(3, 1)`: **1**

4

Nella classe `Intervallo`, definire il metodo `appartiene(int n)` che riporta un valore booleano per indicare se il numero intero dato appartiene agli insiemi disgiunti `[a, b]` e `[c, d]`, dove i valori `a`, `b`, `c` e `d` sono definiti costanti.

```
public static boolean appartiene(int n) {
     return (n >= a && n <= b || n >= c && n <= d);
```

5



L'immagine rappresenta lo stato dello stack delle chiamate dei metodi e dello heap durante l'esecuzione di un programma che manipola una lista di numeri interi ordinata in senso decrescente, implementata mediante la seguente classe `ListaOrdinata` con la classe interna `NodoLista`:

```
public class ListaOrdinata {
    private NodoLista inizio, fine;

    private static class NodoLista {
       Comparable dato;
       NodoLista pros;
    }    ……
}
```

La variabile `lista` sullo stack e' il riferimento alla lista, che nell'immagine viene visualizzata dopo un certo numero di inserimenti. Le due variabili `p` e `q` sono usate come indici di scorrimento della lista per poter effettuare inserimenti e ricerche.

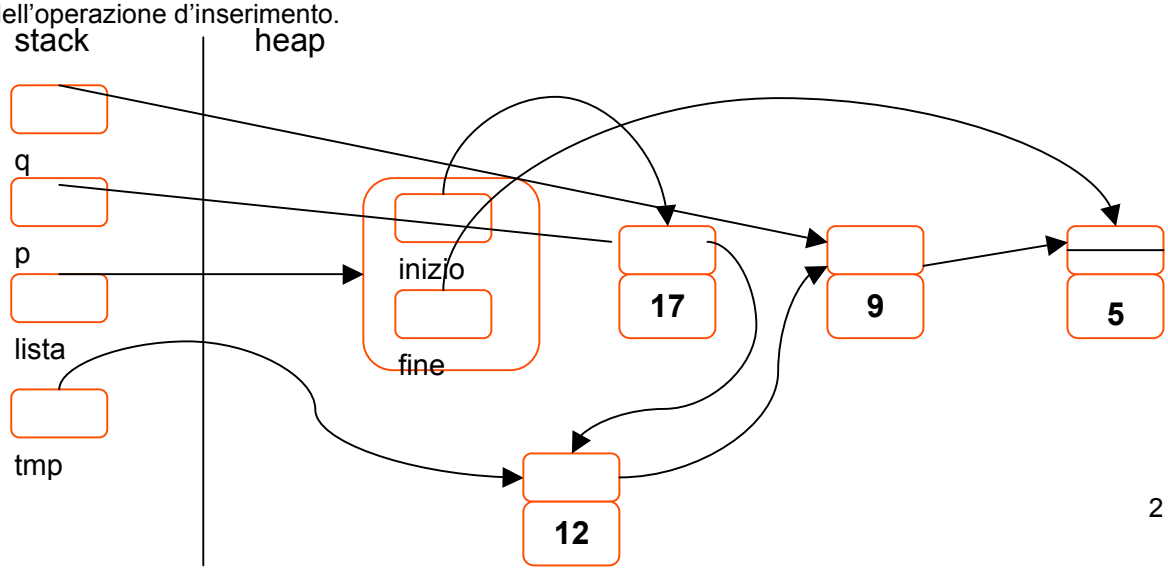
Definite e inizializzate opportunamente, come suggerito dalla immagine, le due variabili `p` e `q`

```
NodoLista p = null; NodoLista q = lista.inizio;
```

Scrivete le due istruzioni per far avanzare di una posizione gli indici sulla lista:

```
p = q;     q = q.pros;
```

Disegnate inoltre la nuova struttura lista dopo aver inserito il nodo con il dato `12` e lo stato delle variabili al termine dell'operazione d'inserimento.



6

Considerate il metodo ricorsivo `palindrome` che riceve in input una stringa e produce in output un valore di verità per indicare se la stringa è una palindrome (si dice palindrome una parola che può essere letta da sinistra a destra o da destra a sinistra).

```
public static boolean palindrome (String s){
  int lun = s.length();
  if (lun <= 1)
        return true;
  else if (s.charAt(0) != s.charAt(lun-1))
        return false;
  else return palindrome (s.substring(1,lun-1)); // R indirizzo di rientro }
```

Tracciare l'andamento dello stack delle chiamate quando viene invocato `palindrome("esse")` in un metodo main all'indirizzo **x**;

stack ⇓ 2

record palindrome par **vuota** lun **0** val di ritorno **true** indirizzo rientro		
record palindrome par **ss** lun **2** val di ritorno **true** indirizzo rientro **R**		
record palindrome par **esse** lun **4** val di ritorno **true** indirizzo rientro **R**		
record main … val di ritorno **true** indirizzo rientro **X**		

stack ⇑ 1 stack ⇑ 3

7

Per una classe `Tabelline` che implementa una struttura dati a matrice di dimensione `10 x 10` di interi definite un metodo `calcola()` che riporta la struttura dati a matrice popolata con i dati numerici corretti del calcolo delle tabelline (cioè righe e colonne di un dato indice contengono la tabellina del numero rappresentato dall'indice).

```
public static int[][] calcola(){
     int[][] matrice = new int[10][];
     for (int i = 0; i < matrice.length; i++) {
        matrice[i] = new int[10];
        for (int j = 0; j < matrice[i].length; j++) {
           matrice[i][j] = (i+1) * (j+1);
        }
     }
     return matrice;
  }
```

8
In una implementazione dinamica della struttura dati stack, definita come segue

```
public class Stack  {
   private NodoStack cima;
   private class NodoStack  {
      Object dato;
      NodoStack pros;
   }

   public Stack()  {
      cima = null;
   }
```

```
public void push(Object o)  {
      NodoStack t = new NodoStack();
      t.dato = o;
      t.pros = cima;
      cima = t;
}
```

Implementare la funzione `top()` che accede al valore del nodo in cima allo stack e lo riporta all'ambiente senza modificare la struttura e un'eccezione non controllata `EmptyStackException()`.

```
public Object top() {
   if (cima == null)
      throw new EmptyStackException();
   else
      return cima.dato;
}

public class EmptyStackException extends RuntimeException {
}
```

Si scriva una istruzione `try-catch` in cui si cattura l'eventuale eccezione lanciata dal metodo `top()` nel contesto di un programma in cui sia stata introdotta la variabile `pila`, onde poter recuperare e inserire almeno un nodo. Gli oggetti da inserire nella pila siano di classe `Integer`. Ad esempio nel codice che segue, si effettua l'operazione di eliminazione della cima (`pop()`) se sullo `stack` c'e un dato valore altrimenti, se lo stack e' vuoto, si inserisce quel valore dato.

```
Stack pila = new Stack();
try {
   if (((Integer)pila.top()).intValue()==x)
      pila.pop();
}
catch (EmptyStackException e) {
      pila.push(Integer(x));
}
```

9
Avendo definito `class Settimana implements java.util.Iterator { };` dire se sono lecite le seguenti istruzioni:

a. `Settimana s = new Settimana();`

   `while(s.hasNext()) <computa s.next()>` **SI** NO

b. `Iterator i = new Settimana();` **SI** NO

c. Qual'è il concetto della programmazione a oggetti per cui diversi oggetti che hanno una interfaccia comune rispondono in modo diverso quando un metodo di quell'interfaccia è invocato: **polimorfismo**

d. Casting è appropriato quando si riceve un riferimento a una classe antenato e si sappia che l'oggetto è una particolare sotto-classe e si voglia usare l'oggetto nella sua funzionalità completa. **SI** NO