Programmazione per Bioinformatica Anno Accademico 2013/2014 **Prima Prova Parziale del 19 Febbraio 2014**	*tempo a disposizione: 2 ore* *consegnare solo la 'bella'*

Esercizio 1 [4 punti]

Si consideri la seguente classe `Loop.java` e si risponda alle domande finali.

```
1  class Loop {
2
3    public static void main(String[] args) {
4      int n;
5
6      if (args.length == 0)
7        System.out.println("Occorre passare un valore come argomento all'avvio.");
8      else {
9        n = Integer.parseInt(args[0]);
10       for (int i = 0; i < n; i++) {
11         for (int j = 0; j < n; j++)
12           System.out.print(i * j % n);
13         System.out.println();
14       }
15     }
16   }
17 }
```

1. [1 punto] Si descriva il funzionamento della seguente riga di codice del programma:
   `n = Integer.parseInt(args[0]);`
   In che modo l'utente dovrebbe passare al programma il valore da assegnare a `n`? A cosa serve la chiamata `Integer.parseInt()`?
2. [2 punti] Assumendo che gli venga passato `4` come valore da assegnare a `n`, cosa stampa a video il programma?
3. [1 punto] Che differenza cè tra le chiamate `System.out.println()` e `System.err.println()`? C'è un punto del programma in cui sarebbe stato meglio usare quest'ultima?

Esercizio 2 [8 punti]

Si scriva il programma `SostituisciStringa.java` che chiede all'utente due stringhe `s1` ed `s2` e (solo) nel caso in cui `s1` abbia lunghezza maggiore di `s2` stampa a video la stringa che si ottiene sostituendo la prima parte di `s1` con `s2`.

Esempio: ecco alcune esecuzioni corrette del programma:

```
$ java SostituisciStringa
Inserire la prima stringa: affanna
Inserire la seconda stringa: azz
Stringa risultante: azzanna

$ java SostituisciStringa
Inserire la prima stringa: svantaggio
Inserire la seconda stringa: lingua
Stringa risultante: linguaggio

$ java SostituisciStringa
Inserire la prima stringa: lingua
Inserire la seconda stringa: svantaggio
La prima stringa non e' piu' lunga della seconda.
```

Esercizio 3 [10 punti]

Si completi la classe `Primes` riportata nel seguito implementando il metodo `printPrimes`:

```
1  import java.util.Scanner;
2
3  class Primes {
4
5    public static void main(String[] args) {
6
7      Scanner tastiera = new Scanner(System.in);
8      int sup;
9
10
11     System.out.print("Inserire l'estremo superiore dell'intervallo di ricerca: ");
12     sup = tastiera.nextInt();
13
14     System.out.println("I numeri primi minori (o uguali) di " + sup +" sono: ");
15     printPrimes(sup);
16     System.out.println();
17   }
18
19   public static void printPrimes(int n) {
20     // parte da implementare!
21   }
22
23   // Eventuali metodi ausiliari...
24 }
```

La classe implementa un programma che chiede all'utente l'estremo superiore dell'intervallo di ricerca e stampa a video tutti i numeri primi compresi nell'intervallo (estremo incluso). È possibile definire ulteriori metodi ausiliari.

Esempio: ecco una possibile esecuzione del programma:

```
$ java Primes
Inserire l'estremo superiore dell'intervallo di ricerca: 100
I numeri primi minori (o uguali) di 100 sono:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Esercizio 4 [10 punti]

Si completi la classe `Vettore` riportata nel seguito implementando il metodo `ruotaTerne`:

```
1  import java.util.Scanner;
2
3  class Vettore {
4    public static void main(String[] args) {
5      if (args.length == 0)
6        System.err.println("Serve la lunghezza del vettore come argomento all'avvio.");
7      else {
8        int[] vettore = new int[Integer.parseInt(args[0])];
9
10       Scanner tastiera = new Scanner(System.in);
11
12       System.out.println("Inserimento degli interi che compongono il vettore...");
13       for (int i = 0; i < vettore.length; i++) {
14         System.out.print("Inserire l'intero in posizione " + (i+1) +": ");
15         vettore[i] = tastiera.nextInt();
16       }
17
18            ruotaTerne(vettore);
19
20       System.out.println("\nVettore dopo la rotazione delle terne:");
21         for (int i = 0; i < vettore.length; i++)
22         System.out.println("Elemento " + (i+1) +": " + vettore[i]);
23     }
24   }
25
26   public static void ruotaTerne(int[] v) {
27     // parte da implementare!
28   }
29 }
```

La classe implementa un programma che prende in input un vettore di interi. Tale array viene considerato come una sequenza di terne di interi. Il programma ruota a destra gli elementi di ogni terna. Se in fondo all'array rimangono 1 o 2 elementi, questi non vengono modificati.

Esempio: il vettore `[1 2 3 4 5 6 7 8 9 10 11]` deve diventare `[3 1 2 6 4 5 9 7 8 10 11]`

Appendice:
Java Platform, Standard Edition 7 API Specification for Class `String`

`char charAt(int index)`: Returns the char value at the specified index.

`int compareTo(String anotherString)`: Compares two strings lexicographically.

`int compareToIgnoreCase(String str)`: Compares two strings lexicographically, ignoring case differences.

`String concat(String str)`: Concatenates the specified string to the end of this string. If the length of the argument string is `0`, then this `String` object is returned. Otherwise, a new `String` object is created, representing a character sequence that is the concatenation of the character sequence represented by this `String` object and the character sequence represented by the argument string.

`boolean equals(Object anObject)`: Compares this string to the specified object.

`boolean equalsIgnoreCase(String anotherString)`: Compares this `String` to another `String`, ignoring case considerations.

`int indexOf(int ch)`: Returns the index within this string of the first occurrence of the specified character.

`int indexOf(int ch, int fromIndex)`: Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

`int indexOf(String str)`: Returns the index within this string of the first occurrence of the specified substring.

`int indexOf(String str, int fromIndex)`: Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

`boolean isEmpty()`: Returns true if, and only if, `length()` is `0`.

`int lastIndexOf(int ch)`: Returns the index within this string of the last occurrence of the specified character.

`int lastIndexOf(int ch, int fromIndex)`: Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

`int lastIndexOf(String str)`: Returns the index within this string of the last occurrence of the specified substring.

`int lastIndexOf(String str, int fromIndex)`: Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.

`int length()`: Returns the length of this string.

`String replace(char oldChar, char newChar)`: Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

`String replace(CharSequence target, CharSequence replacement)`: Replaces each substring of this string that matches the literal `target` sequence with the specified literal `replacement` sequence.

`String replaceAll(String regex, String replacement)`: Replaces each substring of this string that matches the `regex` with the given `replacement`.

`String replaceFirst(String regex, String replacement)`: Replaces the first substring of this string that matches the `regex` with the given `replacement`.

`boolean startsWith(String prefix)`: Tests if this string starts with the specified `prefix`.

`boolean startsWith(String prefix, int toffset)`: Tests if the substring of this string beginning at the specified index starts with the specified `prefix`.

`String substring(int beginIndex)`: Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

`String substring(int beginIndex, int endIndex)`: Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex - beginIndex`.

`String toLowerCase()`: Returns the `String` object obtained by converting all of the characters in this `String` to lower case.

`String toUpperCase()`: Returns the `String` object obtained by converting all of the characters in this `String` to upper case.

`String trim()`: Returns a copy of this string, with leading and trailing whitespace omitted.