Esercizi sul modello Runtime di Java e sulla ricorsione
(Fondamenti di Informatica 2 – Walter Didimo)

Esercizio 1

1) Fornisci una definizione induttiva per la funzione: pari(n): **N** → {true,false}; tale funzione restituisce il valore true se il numero n è pari e false se n è dispari.
2) Fornisci una definizione induttiva per la funzione: prodotto(n,m), che calcola il prodotto tra i numeri naturali n ed m.
3) Usando le definizioni dei punti precedenti, definire il codice della classe Numero che ha il seguente scheletro.

```
class Numero{
/* metodo ricorsivo che determina se il naturale passato come parametro è pari o dispari*/
public static boolean pari(int n){…}

/* metodo ricorsivo che calcola il prodotto dei naturali n e m */
public static int prodotto(int n, int m){…}
}
```

4) Mostrare l'evoluzione dei record di attivazione nello stack delle chiamate per il seguente codice:

```
public static void main (String[] args){
     Numero.pari(3);
     Numero.prodotto (2,2);
}
```

Esercizio 2 – Un oggetto della classe SequenzaDiInteri consente di rappresentare una qualunque sequenza finita di interi. La classe SequenzaDiInteri ha il seguente scheletro.

```
class SequenzaDiInteri{
  /* memorizza la sequenza di interi */
  private int[] seq;

  /* costruttore: crea un oggetto che rappresenta la sequenza a */
  public SequenzaDiInteri (int[] a){…}

  /* calcola la somma di tutti gli elementi della sequenza, usando il metodo sommaRic */
  public int somma (){…}

  /* calcola il prodotto di tutti gli elementi della sequenza, usando il metodo prodottoRic */
  public int prodotto (){…}

  /* calcola il massimo tra tutti gli elementi della sequenza, usando il metodo massimoRic */
  public int massimo (){…}

  /* restituisce una descrizione della sequenza*/
  public String toString (){…}

  /* metodo ricorsivo che calcola la somma degli elementi della sequenza dall'indice 0 all'indice i */
  private int sommaRic (int i){…}

  /* metodo ricorsivo che calcola il prodotto degli elementi della sequenza dall'indice 0 all'indice i */
```

```
  private int prodottoRic (int i){…}

  /* metodo ricorsivo che calcola il massimo tra gli elementi della sequenza
  dall'indice 0 all'indice i */
  private int massimoRic (int i){…}
}
```

Scrivere il corpo dei metodi della classe SequenzaDiInteri, adottando implementazioni di metodi ricorsivi laddove specificato dal commento, e definire inoltre una classe di prova che verifica il corretto funzionamento della classe SequenzaDiInteri.

Esercizio 3 – La classe GestioneStringhe ha il seguente scheletro:

```
class GestioneStringhe{

  /* metodo ricorsivo che calcola se una stringa è palindroma (cioè se leggerla
  da sinistra verso destra è uguale a leggerla da destra verso
  sinistra)*/
  public static boolean palindroma (String s){…}


  /* restituisce la frequenza del carattere c nella stringa s usando il metodo
  ricorsivo frequenzaRic */
  public static int frequenza (String s, char c){…}


  /* restituisce l'esito della relazione d'ordine lessicografico sulle stringhe
  s e t usando il metodo ricorsivo confrontaRic */
  public static int confronta (String s, String t){…}

  /* metodo ricorsivo che calcola la frequenza del carattere c nella stringa s,
  dal carattere in posizione p in poi */
  private static int frequenzaRic (String s, char c, int p){…}


  /* metodo ricorsivo che confronta le stringhe s e t,
  dal carattere in posizione p in poi */
  private static int confrontaRic (String s, String t, int p){…}
}
```

Scrivere il corpo dei metodi della classe GestioneStringhe e definire una classe di prova che verifica il corretto funzionamento della classe GestioneStringhe.

Esercizio 4 – Prendendo spunto dai metodi **`frequenza`** e **`confronta`**, realizzare una soluzione più efficiente per il metodo **`palindroma`** evitando di creare nuove stringhe