Problem 1

Write a static functions which takes a String and returns a similar string but in which the first half of the string is swapped with the second (for example, `"abcd"` → `"cdab"`). If the number of characters is odd, the middle one remains intact.

Problem 2

Create class SortedSLL (*sorted singly-linked list*) implementing singly-linked list of ints. The elements of the list are represented by objects of class Node containing data (an int) and the reference to the next such node. The SortedSLL class contains only one method adding elements to the list, addSorted, which adds them in such a way that the nodes are always ordered by values of data that they contain in ascending order. Add a method to print the list, so the following fragment

```
SortedSLL list = new SortedSLL();
list.addSorted(4);
list.addSorted(1);
list.addSorted(6);
list.addSorted(3);
list.show();
```

will print

```
[ 1 3 4 6 ]
```

Problem 3

Create a class Task, objects of which represent tasks to be done. Objects contain private field descr of type String with a description of a task and (also private) field next of type Task with the reference to the next task (or null). In the class, define constructors and methods

```
public Task(String d, Task n)
public Task(String d)
public void setNextTask(Task t)
public void printTasks()
public static void printTasks(Task head)
public void printTasksRev()
public static void printTasksRev(Task head)
```

where

- the first constructor takes a description and a reference to the next task;
- the second takes only a description — the field next of the object being created is then set to null (this constructor should reuse the previous one!);

- method **setNextTask** sets the field next to the value of the passed reference t;
- method **printTasks** prints in one line descriptions of *this* task and of all the tasks that follow it;
- *static* function **printTasks** prints in one line descriptions of the task represented by head and of all the tasks that follow it (the function may reuse the method of the same name that has already been written);
- *recursive* method **printTasksRev** is analogous to **printTasks**, but prints descriptions in the reverse order, from the latest task to the one on which it has been invoked;
- static function **printTasksRev** printing descriptions in the reverse order, from the latest task to the one represented by head (the function may reuse the method of the same name that has already been written).

In a separate class write **main** function which tests the functionality of class Task. For example,

download *QueueOfTasks.java*

```java
public static void main (String[] args) {
    Task t2 = new Task("Wash the dishes!");
    Task t1 = new Task("Walk the dog!",t2);
    t2.setNextTask(new Task("Clean the room!"));
    Task head = new Task("Get rest!",t1);

    head.printTasks();
    System.out.println();
    head.printTasksRev();
    System.out.println();

    System.out.println();

    Task.printTasks(head);
    System.out.println();
    Task.printTasksRev(head);
    System.out.println();
}
```

should print

```
Get rest! Walk the dog! Wash the dishes! Clean the room!
Clean the room! Wash the dishes! Walk the dog! Get rest!

Get rest! Walk the dog! Wash the dishes! Clean the room!
Clean the room! Wash the dishes! Walk the dog! Get rest!
```

Do not use arrays or any other collections!