Reversing arrays in place

Complete the following program that creates a random array $a$ of integers, of a size pretermined by the user and then reverses the order of its entries in place, that is, without using any other auxiliary array. (The code is available from the course webpage)

**Example:** if the array is $a = \{5, 7, 8, 3, 4\}$, then after reversing the order of its entries the result should be $a = \{4, 3, 8, 7, 5\}$. Again, "in place" means that the end result is stored in the same array $a$ and no other array is used in the course of the computation.

```java
public class ArrayReverser {
    public static void main(String[] args) {
        System.out.print("How big do you want your array to be? : ");
        int N = Keyboard.readInt();

        // declare an array of integers of size N

        // fill the array with random integers between 1 and 36 and show the result
        System.out.println("Your array of " + N + " elements is ");
        for (int i = _; _____; ___) {
            ____ = (int)(_______________) + ___; // assign a random number to a[i]
            System.out.print(" + _____"); // Show element i without breaking the line
        }
        System.out.println();

        // REVERSE THE ARRAY. i.e. if N=5 and a = \{5,7,8,3,4\}
        // then, after the loop, a = \{4,3,8,7,5\}
        // do it IN PLACE, that is, without using any other array
        // HINT: do it by interchanging pairs of numbers
        for (int i = _; i <= _____; i++) {
            /* Here goes
            | the meat
            | of the program */
        }
        // Show the final result
        System.out.println("The reversed array is ");
        for (int i =_; _____; i++)
            System.out.print(" + ___");
        System.out.println(); //final line jump
    }
}

Binary Search / Dictionary Search

When trying to find an item inside a sequence of several items of the same kind a natural first approach is to start with the first element in the sequence, check whether it is the wanted one
and if is not move on to the next one, do the same for it and so forth ... This strategy is simple and is guaranteed to succeed (if the item is in the sequence). But, would you use this strategy if asked to find a name in NYC’s phone guide? Your answer should be ”OF COURSE NOT!”. There is something that makes a phone guide, a dictionary and an old-fashioned library catalog easy to search through; the fact that they are sorted!

A common strategy to search for an element in an ordered collection is called Binary Search (a.k.a. lexicographic or dictionary search). To fix ideas we will describe this search method when applied to an array of numbers in increasing order. Let \( a = \{a[0], a[1], a[2], \ldots, a[N-1]\} \) be this array and imagine we want to search for a value \( z \), that is, we want to determine the index \( i \) so that \( a[i] = z \). To simplify the exposition we will assume first that \( z \) is in the array. Thus, the only thing we know initially is that \( 0 \leq i < N \). Or that \( i \) is in the search window \([L, R]\) determined by the left and right limits \( L = 0 \) and \( R = N \), respectively. The first step in the search is to examine the value of \( a[N/2] \). If \( a[N/2] = z \), we can stop right away and return \( i = N/2 \). If \( z < a[N/2] \) we can conclude that \( i \) is in the new window \([L', R']\) determined by between \( L' = 0 \) and \( R' = N/2 \). If \( z > a[N/2] \) we can conclude that \( i \) is in the window determined by \( L' = N/2 \) and \( R' = N \). Thus, if we don’t find \( z \) the length of the search window is reduced by (roughly) a half. To reduce it further, we can now examine the value of \( a[(L+R)/2] \) by comparing it with \( z \). If \( z = a[(L+R)/2] \), we stop and return \( i = (L+R)/2 \). If \( z < a[(L+R)/2] \) we conclude that \( i \) must be between \( L' = L \) and \( R' = (L+R)/2 \) and if \( z > a[(L+R)/2] \) we conclude that \( i \) must be between \( L' = (L+R)/2 \) and \( R \). It is not hard to see that if we keep iterating this procedure we will eventually find the index \( i \), for which \( a[i] = z \), i.e. the position of \( z \). 

Download the code BinSearch.java from the course website, read it through and answer the question in it. Then complete the method BinarySearch to perform the binary search as explained above. Test your program several times, to make sure it works.

```java
// Method to perform Binary Search (to be included in a class)
// Pre-conditions: a is an array of integers in increasing order
// N is the size of a
// z is an integer whose position needs to be determined
// *z appears in a*
// Pos: the index i for which a[i]==z
public static int BinarySearch (int[] a, int N, int z) {
    int i, L, R;
    L =0; R = N; // initial values for the search window

    while (true) {
        // check whether z == a[(L+R)/2]; if it is, return the position of z
        // if not, update L and R accordingly
    }
}
```

If you have time, modify the method implementation so that it doesn’t assume that \( z \) is in the array \( a \); if \( z \) is not in a BinarySearch should return -1.

Finish your preparation for the prelim

Ask your instructor any questions you might have about Java. Do it!

---

\(^1\)Remember than in Java \( N/2 \) will be an integer if \( N \) is