

1. Java Review

Problem 1.1

Write a method `checkModify` that takes as input an integer parameter `n` and performs the following:

- If `n` is positive, the method returns `false`.
- If `n` is negative, the method returns `true` and changes `n` to `|n|`.

Problem 1.2

Write a method that computes the value of a polynomial p at x :

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

The signature of the method is `double evalPol(double A[], int n, double x)`, where A is the array containing the polynomial coefficients (a_0 is stored in $A[0]$, a_1 in $A[1]$, etc.), n is the degree of the polynomial and x is the value of the variable.

Problem 1.3

Trace the following program on the input array: $A = \{2, 5, 1, 0, 7, 3\}$. What does this function do?

```
public int func(int A[], int n){
    int p= A[0];
    int i= 1;
    int j= n-1;
    while(i<j){
        while((A[i] <= p) && (i<j))
            i++;
        while((A[j] > p) && (i<j))
            j--;
        int tmp= A[i];
        A[i]= A[j];
        A[j]= tmp;
    }

    if(A[i] <= p){
        int tmp= A[i];
```

```

        A[i] = A[0];
        A[0] = tmp;
        return i;
    }
    else {
        int tmp = A[i-1];
        A[i-1] = A[0];
        A[0] = tmp;
        return i-1;
    }
}

```

Problem 1.4

You have an array of $2n$ elements consisting only in zeros and ones. They alternate: 1, 0, 1, 0, and so on. You want to get all the 1 elements to the right-hand end, and all the 0 ones to the left-hand end. The only actions you are allowed to make are those that interchange the positions of two neighboring elements. Write the method $f(\text{int } A[], \text{int } n)$ that implements your solution.

■ **Example 1.1** For $n = 4$, the problem is the following: $A = \{1, 0, 1, 0, 1, 0, 1, 0\} \implies A = \{0, 0, 0, 0, 1, 1, 1, 1\}$. ■

Problem 1.5

Write a method `static void rearrange(int[] A, int n)` that takes as input an array `A` of size `n` and rearranges its elements so that all negative numbers precede all positive numbers (do not allocate a new array).

Problem 1.6

Answer the following:

1. What are generics in Java ? What are the advantages of using generics?
2. Write a generic method to exchange the positions of two different elements in an array.
3. Write a generic method for counting the number of times that a given variable occurs in an array.
4. Write the generic method `public static <T> void reverse(T[] A, int n)` that takes as input a generic array and reverse the order of its n first elements.

Problem 1.7

In Java, it is not possible to create an array of a generic class. For instance, the statement `A<T>[] a = new A<T>[n];` results in a compilation error. To solve this problem it is possible to create a simple data structure that provides the same functionalities as an array (access by index). Complete the class below which uses an array to store data (the array is initially empty).

```

public class GArray<T> {
    ...

    public GArray(int size) { // size is the length of the array
    }

    public T get(int i) { // Return the element at position i
    }

    public void set(int i, T e) { // Put e at index i
    }
}

```

```
}
```

Problem 1.8

Consider the following class:

```
public class Pair<T, U> {
    public T first;
    public U second;
    public Pair(T first, U second) {
        this.first = first;
        this.second = second;
    }
}
```

Write the method `public static <T, U> GArray<Pair<T,U>> pair(T[] A, U[] B, int n)`, which creates a `GArray` of pairs from the first `n` elements of `A` and `B` by pairing each element from `A` with the element from `B` at the same index.

Problem 1.9

When writing a generic class or method, it is possible to constrain the generic type to implement a given interface. For instance, a generic sorting method must compare elements, which means that its generic type must be comparable. One way to enforce this is to require the generic type to implement the Java standard interface `Comparable`.


■ **Example 1.2** If we want the class `A` to be comparable, we define it as:

```
public class A implements Comparable<A> {...}
```

Notice that the interface `Comparable` is a generic interface, with the parameter type indicating the type of objects that "this" object may be compared to. In the example, objects of class `A` can only be compared to objects of the same class.

The interface `Comparable` defines a single method `int compareTo(T o)`, which returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the parameter object.

1. Write the method `public static <T extends Comparable<T>> void sort(T[] A, int n, boolean incr)`, which sorts the `n` first elements of `A` in increasing order if `incr` is true, or decreasing order if `incr` is false.
2. Write a method `test` that creates an array of `String` and uses the method `sort` to sort it.

 Notice that when constraining the generic type `T` to be `Comparable`, we use `extends` and not `implements`.

Problem 1.10

Write the method `static <T> int common(T[] A, int n, T[] B, int m)` that takes as input two arrays `A` and `B` of size `n` and `m` respectively. The method returns the number of common elements between the two arrays.

■ **Example 1.3** The two arrays $A = \{2, 4, 1, 2\}$ and $B = \{7, 2, 4, 4, 9\}$ have two elements in common 2 and 4. ■

Problem 1.11

Write the Java method `static <T> int subString(T[] str, int n, T[] tem, int m)` that searches

for the substring `tem` inside `str` and returns the starting index of its first occurrence if it exists, -1 otherwise. The lengths of `str` and `tem` are passed in `n` and `m` respectively.

■ **Example 1.4** Searching for `tem = AAC` inside `str = CFAABAACSAACAA` returns 5, whereas for `tem = AAS` the return value is -1. ■

Problem 1.12

Write the function `static <T> int count(T[] str, int n, T s, T e)` that counts all the substrings of `str` that starts with `s` and ends with `e`. The parameter `n` is the length of `str`.

```
public class SubString{  
    public int count(char str1[], int n, char str2[], int m){  
        ...  
    }  
}
```

■ **Example 1.5** For example, there are four substrings in "CABAAXBYA" that start with A and end with B. ■