**Sol-Problem1:**

**public** **static** **boolean** checkModify(**int** n){

**if** (n > 0)

**return** **false**;

**else** {

n = -n;

**return** **true**;

}

}

}

**Sol-Problem2:**

**public static double evalPol(double A[], int n, double x)**{

**double** pol = A[0];

**int** de = 1;

**for** (**int** i = 1 ; i <= n ; i++){

pol += A[i] \* Math.*pow*(x,de);

de++;

}

**return** pol;

}

**Sol-Problem3:**

**1)**

A *generic type* is a generic class or interface that is parameterized over types.

The advantages:

A type variable is an unqualified identifier.

A class is generic if it declares one or more type variables.

A method is generic if it declares one or more type variables.

**2)**

**public static <T> void exchange(T a[], int i, int j)**{

T temp;

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

**3)**

**public static** <T> **int** counting(T a[], T t){

**int** count = 0;

**for** (**int** i = 0 ; i < a.length ; i++){

**if** (a[i].equals(t))

count++;

}

**return** count;

}

**4)**

**public static** <T> **void** reverse(T[] A, **int** n){

**int** i = 0;

**int** j = n;

T temp;

**while**(i < j){

temp = A[i];

A[i] = A[j];

A[j] = temp;

i++;

j--;

}

}

**Sol-Problem4:**

**public** **class** GArray<T> {

**private** T array[];

**public** GArray(**int** size){

array = (T[]) **new** Object[size];

}

**public** T get(**int** i){

**if**(i<0||i>array.length)

**return** **null**;

**return** array[i];

}

**public** **void** set(**int** i , T e){

**if**(i<0||i>array.length)

**return** **null**;

array[i] = e;

}

}

**Sol-Problem5:**

**public** **class** Pair<T, U>{

**public** T first;

**public** U second;

**public** Pair(T first, U second){

**this**.first = first;

**this**.second = second;

}

**public** **static** <T, U> GArray<Pair<T, U>> pair(T[] A, U[] B, **int** n){

GArray<Pair<T, U>> g = **new** GArray<>(n);

**for** (**int** i = 0; i < n; i++)

g.set(i, **new** Pair<T, U>(A[i], B[i]));

**return** g;

}

}

**Sol-Problem6:**

**public** **class** Comp {

**public** **static** **void** test(){

String ch[] = {"A","B","D","U","L","R"};

*sort*(ch, ch.length, **false**);

}

**public** **static** <T **extends** Comparable<T>> **void** sort(T[] A, **int** n, **boolean** incr) {

T temp;

**for** (**int** i = 0; i < n - 1; i++){

**for** (**int** j = i + 1; j < n; j++){

**if** (incr) {

**if** (A[i].compareTo(A[j]) > 0){

temp = A[i];

A[i] = A[j];

A[j] = temp;

}

}

**else**{

**if** (A[i].compareTo(A[j]) < 0){

temp = A[i];

A[i] = A[j];

A[j] = temp;

}

}}}

}

}