String operando = [\\d+](file:///\\d+)”;

String operatore = “[\\+\\-\\\*/]”;

String rpn = operando + “(“+operando + operatore+”)\*”;

public statici nt pow(int a, int n) {

if (n<0) throw new IllegalArgumentException();

return pot(a,n);

}

public static int pot(int a, int n) {

if (n==0) return 1;

if (n==1) return a;

int p = pot(a,n/2);

p = p\*p;

if (a%2==0) return p;

return a\*p;

}

Public static <T extends Comparable<? Super T>> void heapSort(T[] v) {

ProrityQueue<T> heap = new ProrityQueue <>(v.length);

for (int i=0; i<v.length; i++)

heap.add(v[i]);

for (int i=0; i<v.length; i++)

v[i] = heap.remove();

}

{…

LinkedList<Integer> ll = new LinkedList<>();

…

ListIterator<Integer> li = ll.listIterator(); boolean aggiunto = false;

while (li.hasNext()) {

Integer next = li.next();

if (next.compareTo(x)>0) {

li.add(x); aggiunto = true; break;}

}

If (!aggiunto)

li.add(x);

}

Public void remove(T x) {

If (testa==null) throw new NoSuchElementException();

While (testa.info.compareTo(x)==0)

Testa = testa.next;

testa = remove(testa,x);

}

Private Nodo<T> remove(Nodo<T> testa, T x) {

If (testa == null) return null;

if (testa.info.compareTo(x)<0)

testa.next=remove(testa.next,x);

if (testa.info.compareTo(x)==0)

Testa = remove(testa.next,x);

return testa;

}

Private static class Nodo<E> {

E info; Nodo<E> next; }