Write the recursive static method ***SearchStack***, that takes a Stack **s** and an element **e** and look for the element **e** in the stack. It found return true. Otherwise, return false. **Don't use auxiliary data structures**. s should not change at the end of the method.

The function's signature: public static <T> boolean SearchStack(Stack<T> s, T e)

Write the recursive static method ***CopyStack***, that takes two Stacks **s1** and **s2** and copies all the elements in **s1** into **s2** in the same order. **Don't use auxiliary data structures**. **s1** should not change at the end of the method.

The function's signature: public static <T> void CopyStack(Stack<T> s1, Stack<T> s2)

Write the recursive method ***Power*** that takes two integers (***base*** and ***exponent***) and calculate the ***base*** to the power of ***exponent***.

The function's signature: public static int Power(int base, int exponent)

Example: Power(2, 4) is 16.

Write the recursive method ***search*** member of the class Linkedlist. That search for an element e and return true if found. False otherwise. **Don't use auxiliary data structures and don't call any of the LinkedList methods.**

The function's signature: public Boolean search(T e)

Write the static recursive method ***SearchList***. That search for an element e in a List ***l*** and return true if found. False otherwise. **Don't use auxiliary data structures.**

The function's signature: public static <T> boolean SearchList(List<T> l, T e)

Write the static recursive method ***PrintQueue***. That prints the elements of the Queue q. **Don't use auxiliary data structures**. q should not change at the end of the method.

The function's signature: public static <T> void PrintQueue(Queue<T> q)

Write the static recursive method ***ReversePrintQueue***. That prints the elements of the Queue q in reverse order. **Don't use auxiliary data structures**. q should not change at the end of the method.

The function's signature: public static <T> void ReversePrintQueue(Queue<T> q)

Write the static recursive method ***ReverseQueue***. That changes the order of the elements in Queue **q** and put them in reverse order. **Don't use auxiliary data structures**.

Write a static method ***replace*** (user of ADT) that takes as input a stack st and two elements x and y. The method replaces all the occurrences of the element x in st with y.

The function's signature: public static<T> void replace (Stack<T> st, T x, T y)

Write a static method ***insertAfter*** (user of ADT) that takes a stack **st**, an index **i**, and an element **e** as inputs. It should insert the element e after the element at position **i** in the stack **st**. You can assume **i** is within the range of the stack, and that the top element has an index of 0.

The function's signature: public static<T> void insertAfter(Stack<T> st, int i, T e)

Write the static method ***removeLast*** (user of ADT) that takes a stack **st** as input, and removes the last element of **st**.

The function's signature: public static<T> void removeLast(Stack<T> st)

Write the method PrintQueue part of the ArrayQueue ADT.