Write a function called rearrangeStack to be outside the ArrayStack class and has two parameters st1 and st2. st1 has data of type E and st2 has data of type character ‘a’ or ‘b’ values. Assume st1 and st2 are of the same size. st1 will be rearranged in a way that all elements corresponding to a’s in st2 will be on top of st1 stack and those correspond to b’s in st2 will be on the bottom of the st1. The function does not return any value.

Public void rearrangeStack(ArrayStack <E> st1, ArrayStack<Character> st2)

{

ArrayStack<Character> st2Copy = new ArrayStack<Character>();

ArrayStack<E> sta = new ArrayStack<E>();

ArrayStack<E> stb = new ArrayStack<E>();

while (!st1.isEmpty())

{

if(st2Copy.pop() == ‘a’)

sta.push(st1.pop());

else

stb.push(st1.pop());

}

while(!stb.isEmpty())

st1.push(stb.pop());

while(! sta.isEmpty())

st1.push(sta.pop());

}

Write a function called insertNode which to be considered within the KWLinkedList. The function accepts one parameter item of type integer and returns false if “this” list (the list) is empty or has less than two nodes. The function inserts a new node before the last node containing item as its data if the summation of the first and last node data are less than item and return true, otherwise, it returns false.

public boolean insertNode(int item)

{

if (size < 2)

return false;

if ((Integer(header.data) + Integer (tail.data)) < item)

{

Node<E> newNode = new Node<E> (item);

newNode.next = tail;

newNode.prev = tail.prev;

tail.prev.next = newNode;

tail.prev = newNode;

size++;

return true;

}

else return false;

}

Write a function called changeCorresponding that has two parameters l1 of type KWLinkedList of data type character and item of type E. l1 and “this” lists are of the same size. If l1 or “this” list is empty, the function returns false. The function looks for the first ‘x’ in l1 and changes the data of the corresponding node in “this” list to item. If change occurs, the function returns true, otherwise it returns false.

public boolean changeCorresponding(KWLinkedList<Character> l1, E item)

{

Node<E> ptr1,ptr2;

ptr1=l1.l1.head;

ptr2=head;

if(size == 0)

return false;

boolean found=false;

while(! Ptr1==null && !found)

{

if(ptr1.data == ‘x’)

found = true;

else

{

ptr1=ptr1.next;

ptr2=ptr2.next;

}

}

if(found)

{ ptr2.data = item;

return true;

}

else

return false;

}