```
::::::::::::::
Problem1.java
::::::::::::::
/*
 * Purpose: Lab 8 Problem 1
 * Status: Complete and thoroughly tested
 * Last update: 04/01/19
 * Submitted:  04/01/19
 * @author: Donald DeWitt
 * @version: 2019.04.01
 */
import java.io.*;

public class Problem1
{
    public static void main(String args[]) throws IOException
    {
        int i = 0;
        String item;
        ListRA<String> list = new ListRA<String>();

        System.out.println("1. Insert item into list.\n" +
                           "2. Remove item from list.\n" +
                           "3. Get item from list.\n" +
                           "4. Search for a specified item in the list.\n" +
                           "5. Clear list.\n" +
                           "6. Print size and content of list.\n" +
                           "7. Exit program.");

        while(i != 7)
        {
            System.out.println("Choose an option: ");
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in
));
            i = Integer.parseInt(br.readLine());

            switch(i)
            {
            case 1:
                System.out.println("Enter the item: ");
                item = br.readLine().trim();
                System.out.println("Enter the index: ");
                int insertNum = Integer.parseInt(br.readLine());
                if(insertNum > -1 && insertNum <= list.size())
                {
                    list.addRA(insertNum, item);
                    System.out.println("Item added successfully.");
                }
                else
                {
                    System.out.println("Item not added. Enter a number greater tha
n -1 and less than less than the size of the list(" + list.size() + ").");
                }
                break;
            case 2:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to remove: ");
                    int removeNum = Integer.parseInt(br.readLine().trim());
                    if(removeNum > -1 && removeNum < list.size())
                    {
                        item = list.get(removeNum);
                        list.remove(removeNum);
                        System.out.println(item + " was removed from the list.");
                    }
                    else
                    {
                        System.out.println("Item not removed. Enter a number great
er than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 3:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to get: ");
                    int getNum = Integer.parseInt(br.readLine().trim());
                    if(getNum > -1 && getNum < list.size())
                    {
                        System.out.println(list.get(getNum) + " was gotten from th
e list");
                    }
                    else
                    {
                        System.out.println("Item not gotten. Enter a number greate
r than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;

            case 4:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the item to search: ");
                    item = br.readLine().trim();
                    int index = search(item, list);
                    if(index != -1)
                    {
                        System.out.println("Successful search. Item " + item + " l
ocated in position " + index);
                    }
                    else
                    {
                        System.out.println("Unsuccessful search. Item not found in
 the list.");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;

            case 5:
                if(!(list.isEmpty()))
                {
```

```
                    list.removeAll();
                    System.out.println("List was cleared of all items.");
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 6:
                if(!(list.isEmpty()))
                {
                    System.out.println("List size: " + list.size() + "\n"
                                       + "Contents: " + list.toString());
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 7:
                System.out.println("Exiting program.");
                break;
            }
        }
    }

    private static int search(String item, ListRA<String> list)
    {
        boolean found = false;
        int index = 0;
        for(int size = list.size(); index < size && !found; index++)
        {
            if(item.equals(list.get(index)))
            {
                found = true;
            }
        }
        if(!found)
        {
            index = -1;
        }
        return index;
    }
}
::::::::::::::
Problem2.java
::::::::::::::
/*
 * Purpose: Lab 8 Problem 2
 * Status: Complete and thoroughly tested
 * Last update: 04/01/19
 * Submitted:  04/01/19
 * @author: Donald DeWitt
 * @version: 2019.04.01
 */
import java.io.*;

public class Problem2
{

    public static void main(String args[]) throws IOException
    {
        int i = 0;
        String item;
        ListRA<String> list = new ListRA<String>();

        System.out.println("1. Insert item into ordered list.\n" +
                           "2. Remove item from list.\n" +
                           "3. Get item from list.\n" +
                           "4. Search for a specified item in the list.\n" +
                           "5. Clear list.\n" +
                           "6. Print size and content of list.\n" +
                           "7. Exit program.");

        while(i != 7)
        {
            System.out.println("Choose an option: ");
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in
));
            i = Integer.parseInt(br.readLine());

            switch(i)
            {
            case 1:
                System.out.println("Enter the item: ");
                item = br.readLine().trim();
                boolean added = false;
                if(list.isEmpty())
                {
                    added = true;
                    list.addRA(0, item);
                }
                else
                {
                    int place = search(item, list);
                    if(place < list.size())
                    {
                        if(!(list.get(place).equals(item)))//Make sure item is uni
que
                        {
                            added = true;
                            list.addRA(place, item);
                        }
                    }
                    else//Item at the end of the list
                    {
                        added = true;
                        list.addRA(place, item);
                    }
                }
                if(added)
                {
                    System.out.println("Item added to the list.");
                }
                else
                {
                    System.out.println("Item not added to the list, not unique.");
                }

                break;
            case 2:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to remove: ");
```

```
                    int removeNum = Integer.parseInt(br.readLine().trim());
                    if(removeNum > -1 && removeNum < list.size())
                    {
                        item = list.get(removeNum);
                        list.remove(removeNum);
                        System.out.println(item + " was removed from the list.");
                    }
                    else
                    {
                        System.out.println("Item not removed. Enter a number great
er than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 3:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to get: ");
                    int getNum = Integer.parseInt(br.readLine().trim());
                    if(getNum > -1 && getNum < list.size())
                    {
                        System.out.println(list.get(getNum) + " was gotten from th
e list");
                    }
                    else
                    {
                        System.out.println("Item not gotten. Enter a number greate
r than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;

            case 4:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the item to search: ");
                    item = br.readLine().trim();
                    int index = search(item, list);
                    if(index != list.size())
                    {
                        System.out.println("Successful search. Item " + item + " l
ocated in position " + index);
                    }
                    else
                    {
                        System.out.println("Unsuccessful search. Item not found in
 the list.");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 5:
                if(!(list.isEmpty()))
                {
                    list.removeAll();
                    System.out.println("List was cleared of all items.");
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 6:
                if(!(list.isEmpty()))
                {
                    System.out.println("List size: " + list.size() + "\n"
                                       + "Contents: " + list.toString());
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 7:
                System.out.println("Exiting program.");
                break;
            }
        }
    }

    private static int search(String item, ListRA<String> list)//ModifiedSequentia
lSearch III
    {
        boolean found = false;
        int pos = 0;

        for(int index = 0; index <= list.size() && !found; index++)
        {
            if(index == list.size())
            {
                pos = index;
                found = true;
            }
            else
            {
                if(item.compareTo(list.get(index)) > 0)
                {
                    if(index+1 < list.size())
                    {
                        if(item.compareTo(list.get(index+1)) < 0)
                        {
                            pos = index+1;
                            found = true;
                        }
                    }
                }
                else if(item.compareTo(list.get(index)) == 0)
                {
                    pos = index;
                    found = true;
                }
            }
```

```
        }
        return pos;
    }
}
::::::::::::::
Problem3.java
::::::::::::::
/*
 * Purpose: Lab 8 Problem 3
 * Status: Incomplete
 * Last update: 04/01/19
 * Submitted:  04/01/19
 * @author: Donald DeWitt
 * @version: 2019.04.01
 */
import java.io.*;

public class Problem3
{
    public static void main(String args[]) throws IOException
    {
        int i = 0;
        String item;
        AscendinglyOrderedStringList list = new AscendinglyOrderedStringList();

        System.out.println("1. Insert specified item into list.\n" +
                           "2. Remove item from list.\n" +
                           "3. Get item from list.\n" +
                           "4. Search for a specified item in the list.\n" +
                           "5. Clear list.\n" +
                           "6. Print size and content of list.\n" +
                           "7. Exit program.");

        while(i != 7)
        {
            System.out.println("Choose an option: ");
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in
));
            i = Integer.parseInt(br.readLine());

            switch(i)
            {
            case 1:
                System.out.println("Enter the item: ");
                item = br.readLine().trim();
                System.out.println(item);
                list.add(item);
                break;
            case 2:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to remove: ");
                    int removeNum = Integer.parseInt(br.readLine().trim());
                    if(removeNum > -1 && removeNum < list.size())
                    {
                        item = list.get(removeNum);
                        list.remove(removeNum);
                        System.out.println(item + " was removed from the list.");
                    }
                    else
                    {
                        System.out.println("Item not removed. Enter a number great
er than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 3:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the index of the item to get: ");
                    int getNum = Integer.parseInt(br.readLine().trim());
                    if(getNum > -1 && getNum < list.size())
                    {
                        System.out.println(list.get(getNum) + " was gotten from th
e list");
                    }
                    else
                    {
                        System.out.println("Item not gotten. Enter a number greate
r than -1 and less than less than the size of the list(" + list.size() + ").");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;

            case 4:
                if(!(list.isEmpty()))
                {
                    System.out.println("Enter the item to search: ");
                    item = br.readLine().trim();
                    int index = search(item, list);
                    if(index != -1)
                    {
                        System.out.println("Successful search. Item " + item + " l
ocated in position " + index);
                    }
                    else
                    {
                        System.out.println("Unsuccessful search. Item not found in
 the list.");
                    }
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;

            case 5:
                if(!(list.isEmpty()))
                {
                    list.clear();
                    System.out.println("List was cleared of all items.");
                }
                else
                {
                    System.out.println("List is empty.");
```

```
                }
                break;
            case 6:
                if(!(list.isEmpty()))
                {
                    System.out.println("List size: " + list.size() + "\n"
                                       + "Contents: " + list.toString());
                }
                else
                {
                    System.out.println("List is empty.");
                }
                break;
            case 7:
                System.out.println("Exiting program.");
                break;
            }
        }
    }

    private static int search(String item, AscendinglyOrderedStringList list)
    {
        boolean found = false;
        int index = 0;
        for(int size = list.size(); index < size && !found; index++)
        {
            if(item.equals(list.get(index)))
            {
                found = true;
            }
        }
        if(!found)
        {
            index = -1;
        }
        return index;
    }
}
:::::::::::::::
AscendinglyOrderedStringList.java
:::::::::::::::
/*
 * Purpose: Lab 8 Problem 3
 * Status: Incomplete
 * Last update: 04/01/19
 * Submitted:  04/01/19
 * @author: Donald DeWitt
 * @version: 2019.04.01
 */
public class AscendinglyOrderedStringList implements AscendinglyOrderedStringListI
nterface
{
    protected String []items;  // an array of list items
    protected int numItems;  // number of items in list

    AscendinglyOrderedStringList()
    {
        items = new String[3];
        numItems = 0;
    }

    public boolean isEmpty()
    {
        return(numItems == 0);
    }

    public int size()
    {
        return numItems;
    }

    public void add(String item) throws ListIndexOutOfBoundsException
    {
        //resize
        int place = search(item);
        boolean add = false;
        if(place < numItems)
        {
            if(!(items[place].equals(item)))
            {
                add = true;
            }
        }
        else if(place == numItems)
        {
            add = true;
        }
        if(add)
        {
            if(numItems+1 > items.length)
            {
                String []bufferList = new String[numItems+1];
                for(int pos = 0; pos < numItems; pos++)
                {
                    bufferList[pos] = items[pos];
                }
                items = new String[items.length + items.length];
                for(int pos = 0; pos < numItems; pos++)
                {
                    items[pos] = bufferList[pos];
                }
            }
            //add and sort
            if(numItems>0)
            {
                items[search(item)] = item;
            }
            else
            {
                items[0] = item;
            }
            numItems++;
        }
    }

    public String get(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            return items[index];
        }
        else
        {
```

```
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on get");
        }  // end if
    } // end get

    public void remove(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            for (int pos = index+1; pos < numItems; pos++)
            {
                items[pos-1] = items[pos];
            }  // end for
            numItems--;
            items[numItems] = null;
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on remove");
        }  // end if
    } //end remove

    public void clear()
    {
        items = (String[]) new Object[3];
        numItems = 0;
    }

    public int search(String item)
    {
        boolean found = false;
        int index = 0;//if found
        int low = 0;
        int high = numItems;
        int mid = high/2;
        for(int pos = mid; pos >= low && pos <= high && !found && pos < numItems &
& pos > 0; pos--)
        {
            if(item.compareTo(items[pos]) == 0)
            {
                found = true;
            }
            else if(item.compareTo(items[pos]) > 0)
            {
                low = mid;
                mid = (low+high)/2;
            }
            else
            {
                high = mid;
                mid = (low+high)/2;
            }
            index = pos;
        }
        return index;
    }

    public String toString()
    {
        StringBuilder info = new StringBuilder();
        for(int index = 0; index < numItems; index++)
        {
            info.append(get(index) + " ");
        }
        return info.toString();
    }
}
```