CS 206 EXAM 1
Spring 2021

Problem 1: Reading Java — 20 points (3 parts: 5, 5 and 10 points)

Write the output of the main function of each of the programs

```
public class Q1P1 {
    private int var=5;
    public Q1P1() {}
    public Q1P1(int var) { this.var = var; }
    public int getVar() {
        return var;
    }
    public void setVar(int v) {
        this.var = v;
    }
    public static void main(String[] args) {
        Q1P1 firstA = new Q1P1(22);
        Q1P1 secondA = firstA;
        secondA.setVar(17);
        Q1P1 thirdA = new Q1P1();
        System.out.println(firstA.getVar() + "
" + secondA.getVar() + " " + thirdA.getVar());
    }}
```

ANSWER: 17 17 5

```
public class Q1P2a {
    private String aString;
    private String bString;
    public Q1P2a(String aString, String bString) {
        this.aString = aString;
        bString = bString;
    }
    public String getAstring() {
        return aString;
    }
    public boolean equals(Object o) {
        if (o instanceof Q1P2a) {
            return this.aString.equals(((Q1P2a)
o).getAstring());
        } else {
            return super.equals(o);
        }}
    public String toString() {
        return "A:" + aString + "\nB:" + bString;
    }

    public static void main(String[] args) {
        Q1P2a ob1 = new Q1P2a("A", "B");
        Q1P2a ob2 = new Q1P2a("B", "C");
        Q1P2a ob3 = new Q1P2a("A", "D");
        System.out.println(ob1 == ob2);
        System.out.println(ob3);
        System.out.println(ob3.equals(ob1));
    }}
```

ANSWER:
false
A:A
B:null
true

```
import java.util.ArrayList;
public class Q1P3a {
    private class P {
        protected int ip=5;
        public P() {}
        public P(int v) { ip=v;}
        public String toString() { return "P:"+ip; }
    }
    private class R extends P {
        protected int ir=10;
        public R(int v) {
            ir = v;
            ip = ir - 10;}
        public String toString() { return "R:"+ip; }
    }
    private class S extends P {
        protected int is=20;
        public S(int v) { super(v); }
        public String toString() { return "S:"+is +"
"+super.toString(); }
    }

    public void doo() {
        ArrayList<P> ph = new ArrayList<>();
        ph.add(new P(1));
        ph.add(new R(2));
        ph.add(new S(3));
        for (int i=ph.size()-1; i>=0; i--) {
            System.out.println(ph.get(i));
        }
    }
    public static void main(String[] args) {
        new Q1P3a().doo();
    }
}
```

ANSWER:

S:20 P:3

R:-8

P:1

Problem 2: Writing Java — 20 points

Write a method to compute the union of two arrays, both of which contain Objects of type String. (You just need to write the union method, not a whole class.) A union contains all of the elements in either the arrays, without any duplications. The signature of the method should be

```
public ArrayList<String> union(String[] array1, String[] array2)
```

The input arrays are proper sets; they contain no duplications. You should assume that there are duplications between the arrays. The returned ArrayList should contain the union.

What is the algorithmic complexity of your union method?

O(n*n)

```
public class Q2 {

    /**
     * Return true iff the given string is NOT in the array list
     * @param arrl the array list to check
     * @param str the string to look for
     * @return
     */
    private boolean notInTheSet(ArrayList<String> arrl, String
str) {
        boolean notIn = true;
        for (int i = 0; i < arrl.size(); i++) {
            if (arrl.get(i).equals(str)) {
                notIn = false;
            }
        }
        return notIn;
    }

    public ArrayList<String> union(String[] arr1, String[] arr2)
{
        ArrayList<String> ret = new ArrayList<>();
        for (int i = 0; i < arr1.length; i++) {
```

```
            // could check for duplicates, but unnecessary per
problem sattement
            ret.add(arr1[i]);
        }
        for (int i = 0; i < arr2.length; i++) {
            if (notInTheSet(ret, arr2[i])) {
                ret.add(arr2[i]);
            }
        }
        return ret;
    }

    public ArrayList<String> intersect(String[] array,
ArrayList<String> aList) {
        ArrayList<String> rtn = new ArrayList<>();
        for (int i=0; i<array.length; i++) {
            for (int j=0; j<aList.size(); j++) {
                if (array[i].equals(aList.get(j))) {
                    rtn.add(array[i]);
                    break;
                }
            }
        }
        return rtn;
    }

    public static void main(String[] args) {
        String[] a = { "A", "S", "D", "F" };
        String[] a2 = { "A", "S", "G", "F" };
        ArrayList<String> b = new ArrayList<>();
        b.add("G");
        b.add("D");
        b.add("G");
        b.add("D");
        ArrayList<String> rtn = new Q2().intersect(a, b);
        System.out.println(rtn);
        System.out.println(new Q2().union(a, a2));

    }
}
```

Problem 3: Complexity — 20 points (4 parts, 5 points each)

For each method of the class `Q3` (below) indicate its big-O time complexity in terms of n, where n is equal to the size the input array `arra`

```
public class Q3 {


    public long part1(int[] arra) {
        long res = 0;
        for (int i=(arra.length*8675309); i>0; i--)
            res += arra[i % arra.length];
        return res;
    }
```

O(n) – multiplying by 8675309 (Jenny!) has no effect on complexity

```
    public long part2(int[] arra) {
        long res=1;
        for (int i=0; i<(arra.length*arra.length); i++) {
            res = res + (long)Math.sqrt(i);
        }
        return res;
    }
```

O(n*n) One loop, but the number of times around is n*n

```
public long part3(int[] arra) {
   long res=0;
   for (int j=0; j<200; j++) {
       for (int i=2000; i<arra.length; i=i+1) {
           res += arra[i];
       }}
   return res;
}
```

O(n) 2 loops, but the outer loop is a constant. This is essentially identical to the first one

```
    public long part4(int[] arra) {
        long res=0;
        for (int i=0; i<arra.length; i++)
            for (int k = 0; k < 7899; k++)
                for (int m = 0; m < 2 * arra.length; m++) {
                    res += (i + k + m);
                }
        return res;
    }
```

O(n*n) – Similar to the previous question, one of the loops is constant. It will slow things down certainly, but not as a function of n

Problem 4: HashTables— 20 points

Suppose you have a hashtable defined to have keys are strings in a special language (defined below). Further suppose the hashtable uses double hashing and is of size 11.

The language is composed of only 3 letter words which are composed of only the letters a, b, c, d.

In the hashing functions those letters translate into the numbers as follows: v(a)=0, v(b)=2, v(c)=3, v(d)=4. The primary hashing function h() is given by:

((v(letter1) + v(letter2) + v(letter3)) * 10) % 11

So for example the hash calculation for "cdd" is:

((v(c) + v(d) + v(d)) *10) % 11
((3 + 4 + 4)*10) %11
(11*10)%11 ==> 0

The secondary hash function is similar

((v(letter1) + v(letter2) + v(letter3)) * 9) % 5

Show the contents of this hashtable after executing all of following operations (you need only show the final contents of the hashtable). Following class convention <"abc", "A"> denotes a key value pair in which "abc" is the key and "a" is the value.

(Continued on next page)

Ad <“aba”, “a”>
Add <“aca”, “b”>
Add <“daa”, “c”>
Add <“baa”, “e”>
Add <“bab”, “f”>
Add <“aba”, “i”>
Add<“aca”, “g”>
Add<“caa”, “h”>

	sum of letter values	mult (sum*10)	h(x) mult%11	mult h2 sum*9	h2(x) multh2 % 5	h2(x)+1
aba	2	20	9	18	3	4
aca	3	30	8	27	2	3
daa	4	40	7	36	1	2
baa	2	20	9	18	3	4
bab	4	40	7	36	1	2
aba	2	20	9	18	3	4
aca	3	30	8	27	2	3
caa	3	30	8	27	2	3

In the above table, the column h2(x) is the second hashing function as printed in the assignment. h2(x)+1 is the change I made during the test. Using this table, the first is action is to put the pair <aba a> into spot 9 in the array. Then <aca b> goes into spot 8 and <daa, c> into 7. <baa, e> would go into 9, but it is occupied, so we go to the h2(x)+1 column and see the step is 4. (9+4)%11 = 2, which is unoccupied, so <baa,e> goes in spot 2. Next <bab,f> wants to go into 7, but is is occupied, h2(x)+1=2, so check (7+2)%11=9, occupied; so next (9+2)%11=0, which is unoccupied, so <bab,f> goes into slot 0. <aba, i> would go into slot 9, which is occupied, but it is occupied by something with key aba, so replace, making the contents of slot 9 <aba, i>. Similarly, <aca, g> would go in slot 8, which is occupied; but by a pair with the key aca so overwrite. Finally, <caa, h> would go into slot 8, which is occupied. So, go to h2(x)+1, the step is 3, so look at (8+3)%11=0, which is occupied, so (0+3)%11 is 3 — unoccupied. So put <caa,h> into 3. See table on next page for final hashtable contents.

	using h2(x)		Using h2(x)+1
0			“<bab, f>
1	“<baa, d>		
2			“<baa, e>
3	<caa, h>		“<caa, h>
4			
5			
6			
7	“<daa, c>		“<daa, c>
8	“~~<aca, b>~~<aca, g>		“~~<aca, b>~~<aca, g>
9	~~“<aba, a>~~<aba, i>		“~~<aba, a>~~<aba, i>
10	“<bab, f>		

Problem 5: Writing Java — 20 points

Write a complete java program that gets string input from a user, one entire line at a time. You may use Scanner for user input as follows:

```
Scanner s = new Scanner();
String line = s.nextLine();
```

With each input line, add the entire line to an ArrayList if either of the following is true

no line of the same length is already in the array list

a line with the same contents is already in the array list.

Keep getting lines from the user until the user enters "99". After receiving "99", stop asking the user for another line and print the contents of the ArrayList.

So, for example, given user input

5
a
5
Qb
99

The program would print (order is not important)

5
5
Qb

```java
/**
 * Answer to Midterm question 5
 * @author gtowell
 * Created: March 2021
 */
import java.util.ArrayList;
import java.util.Scanner;

public class Q5c {

    /**
     * Is a string of the same length already in the ArrayList
     * @param ss -- the array list
     * @param newLine -- the line whose length is to be compared
     * @return true if the array list already contains an item
the same length as newLine
     */
    private boolean newLength(ArrayList<String> ss, String
newLine) {
        boolean newLengthP = true;
        for (int i = 0; i < ss.size(); i++) {
            if (ss.get(i).length() == newLine.length()) {
                newLengthP = false;
            }
        }
        return newLengthP;
    }

    /**
     * Does the array list already contain the same string the
one passed in
     * @param ss -- the array list to be considered
     * @param newLine -- the line to be considered
     * @return true if the array does contain an identical
string.
     */
    private boolean sameString (ArrayList<String> ss, String
newLine) {
        boolean sameStringP = false;
        for (int i = 0; i < ss.size(); i++) {
            if (ss.get(i).equals(newLine)) {
                sameStringP = true;
            }
        }
        return sameStringP;
    }
```

```
    /**
     * Actually do the work of the question.  Get input from the
user and add
     * as needed. The return the filled array list
     */
    ArrayList<String> getInp() {
        Scanner scann = new Scanner(System.in);
        ArrayList<String> ret = new ArrayList<>();
        while (true) {
            String line = scann.nextLine();
            if (line.equals("99")) {
                break;
            }
            if (newLength(ret, line) || sameString(ret, line)) {
                ret.add(line);
            }
        }
        return ret;
    }
    public static void main(String[] args) {
        System.out.println(new Q5c().getInp());
    }
}
```