Please answer questions in the spaces provided. If you make a mistake or for some other reason need more space, please use the back of pages and clearly indicate where the answer can be found. Good luck!
1. (_____/ 20) What gets printed? Please show output as it will appear, or indicate “NO OUTPUT”. If there is an infinite loop, be sure to show some lines of the output followed by “… INFINITE LOOP”.

```java
String word = "Justice";
for (int i = 3; i < word.length() ; i++)
{
    System.out.println(word.charAt(i));
}
```

```
int a = 0;
while (a < 10)
{
    System.out.println(a * 2);
    a += 5;
}
```

```
int b = 8;
do
{
    b--;
    System.out.println(b);
} while (b < 8);
```

```
for (int size = 0; size < 7; size++)
{
    System.out.print(size + " => ");
    // CAREFUL! missing break statements!
    switch(size)
    {
    case 0:
        System.out.println ("A");
        break;
    case 1:
        System.out.print ("B");
    case 2: case 3:
        System.out.println ("C");
        break;
    default:
        System.out.println ("E");
    }
}
```

```
for (int x = 1; x <= 3; x++)
for (int y = 0; y < x; y++)
{
    System.out.print(x + "\t" + y);
    System.out.println();
}
2. [20] Short answer questions
   a) Suppose you look up a method in the Java API and find this method heading:
      ```
      public String mystery(double what)
      ```
      Fill in the following information about this method:

      Method name: ____________________________
      Method return type: _______________________  
      Required parameters for the method: (how many? of what type(s)?)
      _________________________________________

   b) Given a Random object named `rand`, write a Java expression that produces a value in the range `[min ... max]`, where `min` and `max` are variables of type `int` that have positive values, such that `min < max`.

   c) Consider the following code fragment:
      ```java
      String word = "this is my life";
      String message = "";
      int n = 0;
      while (n < 5)
      {
       message = word.charAt(n) + message;
       n++;
      }
      boolean answer = n < 0;
      ```
      What are the values of the following expressions following execution of this code?
      ```
      word.length() ______ word.charAt(1) ______ word.charAt(2) ______
      word.substring(2) ___________ message ____________
      n _______ answer __________ (double)(5 / 10) ________
      (int) (0.48 * 10) __________ (int) 0.48 * 10 ________
      ```
3. [ /20]
Complete the java code below so that it prints a table for the Investment problem: You put an initial amount into a bank account that earns 5% interest per year. Show the yearly returns on your investment until it doubles.
For example, if the user inputs 10000 as the initial amount, a table such as the one shown here would be printed.

<table>
<thead>
<tr>
<th>year</th>
<th>interest</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>$10,000.00</td>
</tr>
<tr>
<td>1</td>
<td>$500.00</td>
<td>$10,500.00</td>
</tr>
<tr>
<td>2</td>
<td>$525.00</td>
<td>$11,025.00</td>
</tr>
<tr>
<td>3</td>
<td>$551.25</td>
<td>$11,576.25</td>
</tr>
<tr>
<td>4</td>
<td>$578.81</td>
<td>$12,155.06</td>
</tr>
</tbody>
</table>
... (keep going until balance >= $20000)

```
import java.text.NumberFormat;
import java.util.Scanner;
public class Investment
{
    public static void main (String[] args)
    {
        Scanner scan = new Scanner(System.in);

        System.out.print("Enter initial balance:");
        double initial = scan.nextDouble();

        int year = 0;
        double rate = 0.05;
        double balance = initial;

        NumberFormat money = NumberFormat.getCurrencyInstance();

        System.out.println("year\tinterest\tbalance");
        System.out.println(year + "\t" + money.format(balance));
    }
}
```
4. [________ / 20]

Construct an algorithm that inputs several positive integers in the range 1...100 from the user, terminated with a -1 (sentinel value). The algorithm should print the numbers entered and determine and print the minimum value. After the minimum is printed, print a goodbye message. You can assume that the numbers entered will be between 1 and 100 (except for the terminating -1), so you do not need to check for mistakes in the input.

Note: Be careful not to process the terminating -1, i.e., make sure your algorithm does not produce -1 as the minimum.

**Example:** If the numbers 25 86 13 54 -1 are entered as input, the algorithm should print:

```
25
86
13
54
Min = 13
Goodbye
```

[Note that the terminating -1 should NOT be printed.]

**Directions:**

Write your algorithm by rearranging and structuring elements chosen from the list below, using indentation to show structure. Do not use anything else and note that not all of these are needed, but you may use one of them more than once, if necessary.

```plaintext
input num
input min
num = 0
min = 0
min = 100
num = 100
num = num + 1
min= num
num = min
if (num < min)
if (num > min)
if (num != -1)
if (num != min)
else
while (num < min)
while (num > min)
while (num != -1)
while (num != min)
print "Min = " min
print "Min = " num
print num
print "Goodbye"
```
5. [________/ 20] Consider an algorithm that repeats the response time experiment that we used in our lab *four times*, while also keeping track of the number of correct answers and computing the average response time of the user:

*Algorithm:*
1. input name
2. print personalized welcome message & instructions

   A. `numCorrect = 0`
   B. `totalTime = 0`

C. *repeat 4 times:*
   3. `a = random int`
   4. `b = random int`
   5. `startTime = current time`
   6. `print question using a, b`
   7. `input answer`
   8. `endTime = current time`
   9. `outcome = 1 or 0 (answer is correct or incorrect)`
   10. `reactionTime = endTime – startTime`
   D. `totalTime = totalTime + reactionTime`
   11. `print outcome as “Correct” or “Incorrect”`
   12. `print reactionTime`
   E. `numCorrect = numCorrect + outcome`

F. `averageTime = totalTime/4`

G. `print averageTime`
H. `print numCorrect`
13. `print goodbye`

The code `ResponseTimeExperiment.java` implementing the *numbered steps* is shown on the next page. Note that the steps labeled by letters are NOT implemented.

Annotate the code for `ResponseTimeExperiment.java` to label all the numbered steps and to show how to incorporate these additional steps, as follows:
- Draw a box around the code corresponding to each numbered step
- Draw a box around the code that needs to go inside the loop C
- *Clearly label all boxes*
- Draw an arrow with a label in the appropriate positions to show where code implementing steps A, B, D, E, F, G, H needs to be inserted.
```java
import java.util.Scanner;
import java.util.Random;
public class ResponseTimeExperiment
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Random rand = new Random();

        System.out.print("Please enter your name: ");
        String name = in.nextLine();

        System.out.println("Hello "+ name
            + ". Please answer as fast as you can."
            + "\\n\\nHit <ENTER> when ready for the question.");

        in.nextLine();  // wait for user to hit <ENTER>
        int a = rand.nextInt(100);
        int b = rand.nextInt(100);
        long startTime = System.currentTimeMillis();
        System.out.print(a + " + " + b + " = ");
        String response = in.nextLine();
        int number = Integer.parseInt(response);
        long endTime = System.currentTimeMillis();
        int outcome = a + b ? 1: 0;
        long reactionTime = endTime - startTime;

        System.out.println(outcome == 1?"Correct!":"Incorrect.");
        System.out.println("time: " + reactionTime + "msec");
        System.out.println("Thank you " + name + ", goodbye.");
    }
```

### Random class

<table>
<thead>
<tr>
<th>double</th>
<th><code>nextDouble()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><code>nextInt(int n)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.</td>
<td></td>
</tr>
</tbody>
</table>

### Math class

<table>
<thead>
<tr>
<th>static double</th>
<th><code>abs(double a)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the absolute value of a double value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static double</th>
<th><code>cos(double a)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the trigonometric cosine of an angle.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static double</th>
<th><code>pow(double a, double b)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the value of the first argument raised to the power of the second argument.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static double</th>
<th><code>random()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a double value greater than or equal to 0.0 and less than 1.0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static long</th>
<th><code>round(double a)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the closest long to the argument.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static double</th>
<th><code>sin(double a)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the trigonometric sine of an angle.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static double</th>
<th><code>sqrt(double a)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the correctly rounded positive square root of a double value.</td>
<td></td>
</tr>
</tbody>
</table>

### String class

<table>
<thead>
<tr>
<th>char</th>
<th><code>charAt(int index)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the char value at the specified index.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><code>compareTo(String anotherString)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Compares two strings lexicographically.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><code>indexOf(int ch)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the index within this string of the first occurrence of the specified character.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th><code>isEmpty()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns true if, and only if, <code>length()</code> is 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><code>length()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the length of this string.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>replace(char oldChar, char newChar)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th><code>startsWith(String prefix)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests if this string starts with the specified prefix.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>substring(int beginIndex)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a new string that is a substring of this string.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>substring(int beginIndex, int endIndex)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a new string that is a substring of this string.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>toLowerCase()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Converts all of the characters in this String to lower case using the rules of the default locale.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th><code>trim()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a copy of the string, with leading and trailing whitespace omitted.</td>
<td></td>
</tr>
</tbody>
</table>