Instructions:

- Keep your eyes on your own paper and do your best to prevent anyone else from seeing your work.
- Do NOT communicate with anyone other than the professor/proctor for ANY reason in ANY language in ANY manner.
- This exam is closed notes, closed books, and no calculator.
- Turn all mobile devices off and put them away now. You cannot have them on your desk.
- Write neatly and clearly indicate your answers. What I cannot read, I will assume to be incorrect.
- Stop writing when told to do so at the end of the exam. I will take 5 points off your exam if I have to tell you multiple times.
- Academic misconduct will not be tolerated. Suspected academic misconduct will be immediately referred to the Emory Honor Council. Penalties for misconduct will be a zero on this exam, an F grade in the course, and/or other disciplinary action that may be applied by the Emory Honor Council.

Time: This exam has 6 questions on 10 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Emory community. I have also read and understand the requirements and policies outlined above.

Signature: ____________________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>6</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (6 points) For each of the following give a basic definition of the term. You do not need to give a formal definition. Then, give an example in java. (An example can be a piece of java code (one or two statements).)

(a) Variable

Solution: identifiers/symbols used to store values for use later in a program.
Example:
int v = 23; (v is variable with int data type)

(b) Casting

Solution: Casting is a unary operator which explicitly converts a datatype to another one.
Example: casting from double to int
int x = (int) 9.16;

(c) Escape character

Solution: a special character (\) that allows Java to change the meaning of the next character.
Example: " (represent the quotation), \n (indicate a new line)

(d) Concatenate

Solution: appending one string to another which results in a single string.
Example: "He" + "llo" => "Hello"
2. (12 points) Evaluate each expression. Then give the result of the evaluation and the data type of the result. If the expression cannot be evaluated or would cause an error, you may simply write "error" for the value. The first row has been done for you.

String s1 = "CS", s2 = "170";
char c1 = 'a', c2 = 't';
int i1 = 6, i2 = 15;
double d1 = 6.0, d2 = 15.0;

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>i1+1</td>
<td>7</td>
<td>int</td>
</tr>
<tr>
<td>i2 % i1</td>
<td>3</td>
<td>int</td>
</tr>
<tr>
<td>i2 / i1 * d1 + c1</td>
<td>109.0</td>
<td>double</td>
</tr>
<tr>
<td>(int) d2 / d1 + i1</td>
<td>8.5</td>
<td>double</td>
</tr>
<tr>
<td>s1.length() + d1</td>
<td>8.0</td>
<td>double</td>
</tr>
<tr>
<td>i1-- == d1</td>
<td>true</td>
<td>boolean</td>
</tr>
<tr>
<td>!(i1 == i2) &amp;&amp; (d1 &lt; d2))</td>
<td>false</td>
<td>boolean</td>
</tr>
<tr>
<td>Integer.parseInt(s2) - d2</td>
<td>155.0</td>
<td>double</td>
</tr>
<tr>
<td>s1.charAt(0) != c1 + 1</td>
<td>true</td>
<td>boolean</td>
</tr>
<tr>
<td>!(c1 &gt; c2) &amp;&amp; (c1 &gt; i2)</td>
<td>true</td>
<td>boolean</td>
</tr>
<tr>
<td>c1 + i1 + s2</td>
<td>&quot;103170&quot;</td>
<td>String</td>
</tr>
<tr>
<td>s1.substring(1,3) + s2.substring(0,3)</td>
<td>Error</td>
<td>IndexOutOfBoundsException</td>
</tr>
<tr>
<td>s1 + &quot;5 + 2.0&quot;</td>
<td>&quot;CS5 + 2.0&quot;</td>
<td>String</td>
</tr>
</tbody>
</table>

Notes: (1) The parenthesis operator has the highest precedence of all operators.
(2) The logical not operator and the casting operator have higher precedence than arithmetic operators, relational/comparison operators, and the other logical operators.
(3) Arithmetic operators have higher precedence than relational/comparison or logical operators.
(4) Relational/comparison operators have higher precedence than logical operators.
(5) Assignment operators have the lowest precedence of all operators.
3. (8 points) Assume the statements below are part of a Java program which compiles and runs. What is the output if the user types 25? Write your answer in the next page.

```java
Scanner myScanner = new Scanner(System.in);
int num = myScanner.nextInt();
boolean check = (num % 2 == 0);

System.out.println("Jan");
if(num >= 30 && check){
    System.out.println("Feb");
} else if (num < 20 || !check){
    System.out.println("March");
} else{
    System.out.println("April");
}

if(num > 25){
    System.out.println("May");
} else{
    if(num > 20){
        System.out.println("June");
    } else{
        if(num < 15){
            System.out.println("July");
        } else{
            System.out.println("Aug");
        }
    }
    System.out.println("Sep");
}

switch(num + '0'){
    case 25:
        System.out.println("One");
        break;
    case 250:
        System.out.println("Two");
        break;
    case 75:
        System.out.println("Three");
        break;
    default:
        System.out.println("Four");
        break;
}

while(num * 2 >= 40){
    System.out.println(num);
    num -= 5;
}
System.out.println(num);
```
Solution:

Jan
March
June
Sep
Four
25
20
15

-1 for each omission or incorrect answer
4. Consider the following program (with the lines numbered for convenient reference).

```
1: import java.util.Scanner;
2: public class LogicalError {
3:     public static void main(String args[]) {
4:         Scanner myScanner = new Scanner(System.in);
5:         String inputStr = myScanner.next();
6:         int i;
7:         char letter;
8:         for(i = 0; i < inputStr.length(); i++){
9:             letter = inputStr.charAt(i);
10:            inputStr = inputStr + letter;
11:            System.out.println(inputStr);
12:        }
13:    }
14: }
15: }
```

The intent of the program is to double the text in inputStr by adding the characters to it one by one. For example for the inputStr as "MSC", the program should print "MSCM", "MSCMS",..., and stop after printing "MSCMSC". However, when we run this program with inputStr as "MSC", we observe the following output (running forever).

```
MSCM 
MSCMS
MSCMASC 
MSCMASCMS
MSCMASCMS (forever)
```

(a) (3 points) Explain why the program runs forever and doesn’t stop after doubling the string.

**Solution:** The error is on line 8. `inputStr.length()` will always be increasing since inputStr is changing inside the for loop (line 10 by concatenating a new letter), so `i` will always be smaller than `inputStr.length()`, making the for-loop condition to be true all the time.
(b) (3 points) What line(s) of code would you change to eliminate this error? Change the line(s) to make the program function correctly, assuming that `inputStr` can be any string.

**Solution:**

One fix is to define another String typed variable as

```java
String outputStr = inputStr;
```

before the loop and then use it inside the loop instead of lines 10 and 11.

```java
outputStr = outputStr + letter;
System.out.println(outputStr);
```

Another fix can be storing the size of `inputStr` in an integer as

```java
int len = inputStr.length();
```

before the loop and then change the condition in line 8 to

```java
for(i = 0; i < len; i++)
```
5. (8 points) The program below accepts 2 integers from the user. You should add code to print out all the common divisors of the two numbers. You may assume the user only enters integer values in the terminal window.

Examples of running this program:

```java
>>> java Divisors
8 12
The common divisors are:
1
2
4
```

```java
import java.util.Scanner;

public class Divisors {
    public static void main(String args[]) {
        Scanner myScanner = new Scanner(System.in);

        int a = myScanner.nextInt();
        int b = myScanner.nextInt();

        /*----------- Your code here -------------*/
    }
}
```

Solution:

One possible solution: (while loop)
```java
int i = 1;
System.out.println("The common divisors are: ‘’");
while( i <= a && i <= b){
    if( (a % i == 0) && (b % i == 0))
        System.out.println(i);
}
/* ************* Your code here ***************/
```

Another possible solution: (for loop)
```java
System.out.println("The common divisors are: ‘’");
for(int i = 1; i <= a && i <= b; i++){
    if((a % i == 0) && (b % i == 0))
        System.out.println(i);
}
```
6. (10 points) You want to write a program that reads in a word from the user and checks if this word has the same number of uppercase A’s and lowercase a’s.

Examples of running this program:

```java
>>> java CheckAa
Enter a word: abaAdn
Not the same number of A’s and a’s!

>>> java CheckAa
Enter a word: AbcAbaa
The same number of A’s and a’s!
```

```java
import java.util.Scanner;

public static CheckAB {
    public static void main(String[] args) {
        Scanner myScanner = new Scanner(System.in);

        System.out.println("Enter a word: ");
        String s = myScanner.next();

        /*------ Your code here -------*/
    }
}
```

Solution:

One possible solution: (while loop)

```java
int i = 0;
int countA = 0, counta = 0;
while (i < s.length()) {
    if(s.charAt(i) == 'A') {
        countA++;
    } else if(s.charAt(i) == 'a') {
        counta++;
    }
    i++;
}
```
if(countA == counta){
    System.out.println("The same number of A’s and a’s!");
} else {
    System.out.println("Not the same number of A’s and a’s!");
}

Another possible solution: (for loop)

int countA = 0, counta = 0;
for (int i = 0; i < s.length(); i++) {
    if(s.charAt(i) == 'A') {
        countA++;
    } else if(s.charAt(i) == 'a') {
        counta++;
    }
}
if(countA == counta){
    System.out.println("The same number of A’s and a’s!");
} else {
    System.out.println("Not the same number of A’s and a’s!");
}