Name (print): ____________________________________________

- **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 8 questions on 10 pages including the title page. Please check to make sure all pages are included. You will have 75 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>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>9</td>
<td>12</td>
<td>7</td>
<td>9</td>
<td>75</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1
1. (10 points) Define 5 of the 6 terms below. You do not need to give a formal definition just a good description. You may include an example if it is helpful to your answer. Be brief, and clearly mark the term you do not want me to grade. Otherwise, I will discard the last term.

(a) compiler

**Solution:** software which translates a high level language to machine language

(b) operator

**Solution:** symbol which causes some sort of computation to occur
Ex: =, (), +, ! etc.

Common issue: Using term to define term. Ex: “An operator causes operations to occur.” This doesn’t clearly demonstrate understanding of what an operator/operation is.

(c) class

**Solution:** syntax in Java for encapsulating your program or a collection of related methods.

(d) variable

**Solution:** identifiers/symbols used to store values for use later in a program
Ex: \( v = 23; \) (\( v \) is variable)

(e) datatype

**Solution:** classification scheme for data in a programming language. Used to specify encoding of bits
Ex: in Java, some datatypes are `int`, `char`, `long` etc

(f) character

**Solution:** number mapped to single symbol in written language
Ex: ‘a’, ‘*’, or ‘7’, etc.

Common issue: using term to define term. Ex: “A character is a value of the character datatype used to represent a character.” While this is true, it never clearly demonstrates an understanding of what a character is.
2. (6 points) Convert the following binary numbers to decimal equivalents
   
   (a) $10111_2$  
   \[23\]  
   
   (b) $1001_2$  
   \[9\]  
   
   (c) $10100_2$  
   \[20\]  

3. (6 points) Convert the following decimal number to their binary representation.
   
   (a) 62  
   \[111110\]  
   
   (b) 125  
   \[1111101\]
4. (16 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 is not proper Java syntax, you may simply write “error” for the value. The first row has been done for you.

```java
String s1 = "Emory", s2 = "University", s3 = ":42";
char c1 = '5', c2 = '*';
int i1 = -3, i2 = 2; i3 = 4
double d1 = 3.5, d2 = 7.0;
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>4+1</td>
<td>5</td>
<td>int</td>
</tr>
<tr>
<td>i1 + i2 / i2</td>
<td>-2</td>
<td>int</td>
</tr>
<tr>
<td>s1 + i1 * i3</td>
<td>Emory-12</td>
<td>String</td>
</tr>
<tr>
<td>i1 * d2</td>
<td>-21.0</td>
<td>double</td>
</tr>
<tr>
<td>d1 / i1 + i2</td>
<td>.83</td>
<td>double</td>
</tr>
<tr>
<td>i2 != d2/d1</td>
<td>false</td>
<td>boolean</td>
</tr>
<tr>
<td>(int)s3 + i2</td>
<td>error</td>
<td>can’t cast strings to ints</td>
</tr>
<tr>
<td>i2 %% i3</td>
<td>2</td>
<td>int</td>
</tr>
<tr>
<td>(int)d2 + d1 + i3</td>
<td>14.5</td>
<td>double</td>
</tr>
<tr>
<td>&quot;&quot; + i3 + c2 + i1</td>
<td>4*-3</td>
<td>String</td>
</tr>
<tr>
<td>i1 + s3 + d2</td>
<td>-3-427.0</td>
<td>String</td>
</tr>
<tr>
<td>d1 &lt; i2*i3</td>
<td></td>
<td>i1 &lt; d2</td>
</tr>
<tr>
<td>s1.charAt(5)</td>
<td>error</td>
<td>index out of range</td>
</tr>
<tr>
<td>s2.charAt(5)</td>
<td>r</td>
<td>char</td>
</tr>
<tr>
<td>!(d1 = d2/i2)</td>
<td>error</td>
<td>assignment vs. equality</td>
</tr>
<tr>
<td>c1 + 2</td>
<td>typo in problem</td>
<td>credit for all</td>
</tr>
</tbody>
</table>

Notes:
- The parenthesis operator has the highest precedence of all operators.
- The logical not operator and the casting operator have higher precedence than arithmetic operators, relational/comparison operators, and the other logical operators.
- Arithmetic operators have higher precedence than relational/comparison or logical operators.
- Relational/comparison operators have higher precedence than logical operators.
- Assignment operators have the lowest precedence of all operators.
5. (9 points) Assume the statements below are part of a Java program which compiles and runs. What is the output if the user types 10?

```java
Scanner in = new Scanner(System.in);
int num = in.nextInt();

System.out.println("Summer:");
if (num % 10 == 0) {
    System.out.println("baseball");
} else {
    System.out.println("sailing");
} if (num % 2 == 5) {
    System.out.println("golf");
} if (num % 5 == 0) {
    System.out.println("track and field");
} else {
    System.out.println("swimming");
}

System.out.println("Fall:");
if(num <= 10) {
    System.out.println("soccer");
} if (num == 10) {
    System.out.println("basketball");
} else if (num <= 10) {
    System.out.println("football");
} else {
    System.out.println("volleyball");
}

System.out.println("Winter:");
if (num > 10) {
    System.out.println("skiing");
    if (num > 20) {
        System.out.println("ice skating");
    } else {
        System.out.println("bobsled");
    }
} else {
    System.out.println("curling");
    if (num != 5) {
        System.out.println("hockey");
    } else {
        System.out.println("luge");
    }
}
```

Solution: 5
<table>
<thead>
<tr>
<th>Season</th>
<th>Sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>baseball, track and field</td>
</tr>
<tr>
<td>Fall</td>
<td>soccer, basketball</td>
</tr>
<tr>
<td>Winter</td>
<td>curling, hockey</td>
</tr>
</tbody>
</table>

- .5 for each omission or incorrect inclusion (not including “Summer,” “Fall,” or “Winter”)
6. For each of the code fragments below, give the output. If the code results in an infinite loop, write the first few outputs, and then indicate that it is an infinite loop.

(a) (4 points)
```java
int x = 1
while (x < 5) {
    System.out.println(x);
    if (x == 3) {
        x++;
    } else {
        x+=2;
    }
}
```

Solution: 1
3
4

(b) (4 points)
```java
for(int x = 2; x <= 9; x+=2) {
    System.out.println(x);
    if (x % 2 == 1) {
        x--;
    }
}
```

Solution: 2
4
6
8
(Note that loop increments by 2 each time!)

(c) (4 points)
```java
for (int x = 3; x != 0; x--) {
    System.out.println(x);
}
```

Solution: 3
2
1
7. (7 points) Complete the program below. It should prompt the user to enter 100 integers, and compute the sum of those integers. Then it prints the sum to the screen. *Hint: Use a loop.*

```java
import java.util.Scanner;

public class SumInts{
    public static void main (String[] args){
        Scanner in = new Scanner(System.in);
        /*----------- Your code here -------------*/
    }
}
```

**Solution:**

```java
int sum=0;
for (int i=0; i < 100; i++) {
    System.out.print("Enter an integer: ");
    sum += in.nextInt();
}
System.out.println("The sum is " + sum);
```

or

```java
int sum = 0;
int count = 0;
while (count < 100) {
    System.out.print("Enter an integer: ");
    sum += in.nextInt();
}
System.out.println("The sum is " + sum);
```

**Scoring:**
+3 reads in 100 integers
+3 keeps track of sum correctly
+1 prints output to screen correctly

**Common problems:** writing loops that executed 101 times.
Eg. `for(int x = 0; x <= 100; x++)`.
Re-declaring `sum` variable inside loop which “resets” variable every time loop executes.
8. (9 points) Complete the program below. We are having a party with amounts of soda and candy. The program print out whether your party is “bad,” “good,” or “great” based on the following rules. A party is “good” if both the amount of soda and the amount of candy are at least 5. However, if either the amount of soda or the amount of candy is at least double the amount of the other one, the party is “great”. However, in all cases, if either the amount of soda or the amount of candy is less than 5, the party is always “bad”.

Examples:

<table>
<thead>
<tr>
<th>Amt. Soda</th>
<th>Amt. Candy</th>
<th>Party Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>“good”</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>“bad”</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>“great”</td>
</tr>
</tbody>
</table>

```java
import java.util.Scanner;

public class Party {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("How much soda? ");
        int soda = in.nextInt();
        System.out.print("How much candy? ");
        int candy = in.nextInt();
        /* -------------------- Your code here ---------------*/
    }
}
```

**Solution:** Many possible solutions. One example:

```java
if (soda < 5 || candy < 5) {
    S.O.P("Bad party");
} else {
    if (soda >= 2*candy || candy >= 2*soda) {
        S.O.P("Great party");
    } else {
        S.O.P("Good party");
    }
}
```

Scoring:
+3: identifying bad party
+3: identifying good party
+3: identifying great party

Common mistakes: writing conditions which caused problems due to the order in which they were written. Ex:

```java
if (candy >= 5 && soda >= 5) {good}
else if (candy >= 2*soda || soda >= 2*candy) {great}
else {bad}
```

The above will never have any great parties as any parties that should be great (eg row 3 in the examples) will meet the condition for a good party first. Additionally, this will not correctly identify all bad parties (eg row 2 in the examples) which will meet the criteria above for a great party.