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 15 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>13</td>
<td>15</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>6</td>
<td>8</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. (13 points) Choose 5 of the 6 errors, and (1) identify the error as a syntax error, a runtime error, or a logical error and (2) give a small piece of code which demonstrates the error. You may label your code for clarity, but you SHOULD NOT write a definition. You DO NOT need to write a complete program. Code which clearly demonstrates or leads to the error is sufficient. Mark the error you do not want me to grade. Otherwise, I will discard the last error.

(a) index out of bounds error

**Solution:** runtime error

```java
String s = "Hello";
S.O.P.(s.charAt(5));
```

(b) infinite while loop

**Solution:** logical error

```java
int x = 4;
while (x != 0) {
    x -= 3;
}
```

(c) cannot find symbol

**Solution:** syntax error

```java
s = "Hello";
//s not previously defined
or something like
//Scanner in previously setup
in.nextInt()
```

(d) possible loss of precision

**Solution:** syntax error

```java
int x = 4.5;
```

(e) unreachable code
Solution: syntax error

```java
while (x > 0) {
    if (x == 4) {
        continue;
        x++; //unreachable code
    }
}
```

(f) infinite for loop

Solution: logical error

```java
for (int x = 4; x > 0; x++) {
    ...
}
```

Solution: See 1.11 (8-9th ed), 1.10 (10th ed)

Scoring:
+1 for error classification (syntax, logical, runtime)
+1.5 for code demonstrating error
2. (15 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 = "Happy", s2 = "Halloween", s3 = "100";
char c1 = ‘1’, c2 = ‘A’;
int i1 = 4, i2 = 10;
double d1 = 1.0, d2 = 5.5, d3 = 0.5;
```

<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>s2.length()</td>
<td>9</td>
<td>int</td>
</tr>
<tr>
<td>s2.charAt(i1+1)</td>
<td>w</td>
<td>char</td>
</tr>
<tr>
<td>Double.parseDouble(s3)</td>
<td>100.0</td>
<td>double</td>
</tr>
<tr>
<td>i1 / i2</td>
<td>0</td>
<td>int</td>
</tr>
<tr>
<td>d1 ** 4 &gt;= i1</td>
<td>error</td>
<td></td>
</tr>
<tr>
<td>s3 + i1 * (int)d2</td>
<td>10020</td>
<td>String</td>
</tr>
<tr>
<td>d2 + i2 + s3</td>
<td>15.5100</td>
<td>String</td>
</tr>
<tr>
<td>s3 + d2 + i1</td>
<td>1005.54</td>
<td>String</td>
</tr>
<tr>
<td>(char)(c2 + 4)</td>
<td>'E'</td>
<td>char</td>
</tr>
<tr>
<td>i2 + c1</td>
<td>59</td>
<td>int</td>
</tr>
<tr>
<td>d2 &lt; i1+i2</td>
<td></td>
<td>d2 == i1+d3*3</td>
</tr>
<tr>
<td>s2 + c1 + c2</td>
<td>Halloween1A</td>
<td>String</td>
</tr>
<tr>
<td>!(d1 + i1 + d3 &gt;= d2)</td>
<td>false</td>
<td>boolean</td>
</tr>
<tr>
<td>i2 &gt; i1 &amp; &amp; !(d2 &gt; d1)</td>
<td>false</td>
<td>boolean</td>
</tr>
<tr>
<td>s1 + '&quot;'&quot; + s3 + '&quot;\&quot;&quot;</td>
<td>Happy&quot;100&quot;</td>
<td>String</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 opera-
tors, 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.
3. (8 points) Assume the statements below are part of a Java program which compiles and runs. What is the output if the user enters 10 and 20 in that order?

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

if(a <= 40 && b >= 40) {
    System.out.println("apple");
} else if (a >= 40 || b >= 40) {
    System.out.println("banana");
} else if (a <= 40 || b >= 40) {
    System.out.println("orange");
} else {
    System.out.println("mango");
}

if (!(a <= 30) && b >= 20) {
    System.out.println("pineapple");
} else {
    System.out.println("pear");
} if (a >= 30 && b >= 20) {
    System.out.println("blueberry");
} else {
    System.out.println("strawberry");
}

switch(a * b) {
    case 200:
        System.out.println("coconut");
    case 150:
        System.out.println("grape");
    case 100:
        System.out.println("peach");
        break;
    default:
        System.out.println("watermelon");
}

if (a == 20 || b == 20) {
    System.out.println("fig");
} if ((b-a) >= a ) {
    System.out.println("raspberry");
} if (a == 10 && b != 10) {
    System.out.println("grapefruit");
} else {
    System.out.println("lime");
}
```
Solution: orange
pear
strawberry
coconut
grape
peach
fig
raspberry
grapefruit
4. 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) (3 points)

```java
for(int i = 3; i <= 10; i++) {
    System.out.println(i);
    i += 2;
}
```

Solution: 3
6
9

(b) (2 points)

```java
int x = 8;
while(x >= 0) {
    System.out.println("x: " + x);
    if (x % 3 == 0) {
        x += 3;
        break;
    } else {
        x--;
        continue;
    }
}
System.out.println("Final value: " + x);
```

Solution: x: 8
x: 7
x: 6
Final value: 9

(c) (2 points)

```java
int i = 0;
while (i < 10) {
    System.out.println(i);
    if (i == 3) {
        continue;
    } else {
        i = i+2;
    }
    i++;
}
```

Solution: 0
3
3
infinite loop
(d) (4 points)
    int a = 1;
    int b = 2;
    while (a < 10) {
        switch(a){
            case 1:
                case 2:
                    System.out.println(a);
                    a++;
                    break;
            case 4:
                System.out.println(a+b);
                a += b;
                case 5:
                    System.out.println(a-b);
                    break;
        }
        if (a % 7 == 0) {
            System.out.println(a);
            break;
        }
        a++;
    }

    Solution: 1
    6
    4
    7
5. Prof. Summet is writing code to identify prime numbers (numbers which are only divisible by 1 and themselves). For example: 49 is not a prime number (divisible by 7) while 53 is a prime number. However, her code isn’t working as expected!

```java
import java.util.Scanner;

public class Primes{
    public static void main (String[] args){
        Scanner in = new Scanner(System.in);
        int num = in.nextInt();
        boolean isPrime = true;

        for(int i = 1; i <= num; i++) {
            if (num % i == 0) {
                isPrime = false;
                break;
            }
        }

        String s = num + " is ";
        if (!isPrime) {
            s = s + "not ";
        }
        s += "prime."
        System.out.println(s);
    }
}
```

(a) (1 point) As currently written, what will the code output for the number 4?

**Solution:** 4 is not a prime number

(b) (1 point) As currently written, what will the code output for the number 5?

**Solution:** 5 is not a prime number

(c) (3 points) Explain Prof. Summet’s logical error(s) and correct it/them. You do not need to rewrite the entire program, only the problematic lines of code.

**Solution:** The bounds on the loop are incorrect. Since all numbers (not just primes) are divisible by 1, Java will always calculate that num % 1 == 0 is true. The loop will then terminate via the break; statement. To correct it, we should change the loop on line 9 to for (int i = 2; i < num; i++). Note that you also must change the continuation condition to < not <=. Otherwise, you will see the same result (all numbers reported to be prime) since all numbers are divisible by themselves.
6. (6 points) We are writing a program to help us play blackjack. The program should allow the user to enter 2 integers. It should tell us what number is closest to 21 without going over. However, if both numbers are over 21, the program should tell us that we “busted”.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>19, 21</td>
<td>21</td>
</tr>
<tr>
<td>21, 19</td>
<td>21</td>
</tr>
<tr>
<td>19, 22</td>
<td>19</td>
</tr>
</tbody>
</table>

```java
class BlackJack {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a, b;
        /***************Your code here*******************/
    }
}
```
Solution:

```c
if (a > 21 && b > 21)
    S.O.P("busted");
else if (a > 21)
    S.O.P(b);
else if (b > 21)
    S.O.P(a);
else if (a > b)
    S.O.P(a
else
    S.O.P(b)
```

Scoring:
+1: for number inputs via scanner
+1: case where both greater than 21 (“busted”)  
+1: case where a lte 21 and b gt 21 (a)  
+1: case where b lte 21 and a gt 21 (b)  
+1: case where a gt b and both lte 21 (a)  
+1: case where b gt a and both lte 21 (b)

Common error: Not handling case where a gt 21 but b lte 21 or vice versa.
7. (8 points) Complete the program below. The program should read in a String from the user. If the
string has an even number of characters, the program should print out the even-indexed characters. If
the string has an odd number of characters, the program should print out the odd-indexed characters.
Examples are below:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello CS 170</td>
<td>HloC 7</td>
</tr>
<tr>
<td>Hello CS 170!</td>
<td>el S10</td>
</tr>
</tbody>
</table>

```java
import java.util.Scanner;

public class EvenOdd {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a string: ");
        /* --------------------------- Your code here ---------------------------*/
    }
}
```
Solution:

```java
int start = 0;
if (s.length() % 2 == 1) {
    start = 1;
}

String out = "";
for (int i = start; i < s.length(); i+=2) {
    out = out + s.charAt(i);
}
System.out.println(out)
```

Scoring:
+1: reading in string and storing in variable
+2: identifying odd length strings
+2: identifying even length strings
+2: utilizes loop to select characters correctly
+1: output
8. (9 points) Complete the program below, which searches through a user entered String for the substring “cat” or “dog”. The program should print out the number of times “cat” or “dog” appears in the string. You may assume the string entered is all lowercase. (In other words, you do not need to account for “CAT”, “cAT”, etc.).

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>cattttdogcat</td>
<td>3</td>
</tr>
<tr>
<td>cadocadoca</td>
<td>0</td>
</tr>
<tr>
<td>catdodododo</td>
<td>1</td>
</tr>
</tbody>
</table>

```java
public class CatDog {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter a string: ");
        //*********** Your code here ***********
    }
}
```
Solution:

```java
int count = 0;
for(int i = 0; i <= s.length()-3; i++) {
    char a = s.charAt(i);
    char b = s.charAt(i+1);
    char c = s.charAt(i+2);
    if (a == 'c' && b == 'a' && c == 't') {
        count++;
    }
    if (a == 'd' && b == 'o' && c == 'g') {
        count++;
    }
}
System.out.println(count);
```

Scoring:
+1 String read in via scanner and stored in variable
+3 for loop running appropriate amount of times (termination!)
+2 for counting instances of “cat”
+2 for counting instances of “dog”
+1 output

Common errors:
1) Not terminating the loop correctly. If you have a solution like above and run the loop until i < s.length(), the end iterations of the loop will give you runtime errors when the s.charAt(i+1) and s.charAt(i+2) exceed the length of the String.

2) using the == operator to compare Strings. You can use this operator to compare characters, but you need to use the instance method s1.equals(s2) to compare Strings correctly.