• **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 14 pages including the title page. Please check to make sure all pages are included. You will have 65 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>5</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>17</td>
<td>4</td>
<td>15</td>
<td>8</td>
<td>65</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. For each print statement below, give the resulting value which is printed to the screen. **Your answer should indicate type:** that is, Strings should be enclosed in quotation marks ("), characters should be enclosed in single quotation marks ('), and numerical types should correctly indicate floating point or integer precision. Each statement should be evaluated independently. In other words, assume the statements begin with the initial variable values below. If the statement would result in an error, you may simply write “error” as your answer. There is an ASCII table on the last page of the exam.

```java
String word = "170CS2015";
int i = 15;
int j = 12;

(a) (1 point) System.out.println(word.charAt(word.length()));
   (a) error - Index out of bounds

(b) (1 point) System.out.println(word.charAt(1));
   (b) '7'

(c) (1 point) System.out.println(word.charAt(2) + 3);
   (c) '0' + 3 == 48 + 3 == 51

(d) (1 point) System.out.println(i++ - --j);
   (d) 15-11 == 4

(e) (1 point) System.out.println(++i + j);
   (e) 16+12 == 28
```

2. Code Tracing

(a) (2 points) You invoke a program from the command line with:
```java
java MyProgram Hello 170 2015
```
Give the value and type of `args[2]` in main?

```
(a) 2015, String
```

(b) (2 points) What is returned by the following method if it is invoked with the call `mystery(4)`?
```java
public static int mystery(int n) {
    if(n==0) {
        return 0;
    } else {
        return n + mystery(n-1);
    }
}
```

```
(b) 10
```
3. Consider the following array: \[12 \ 8 \ 4 \ 32 \ 20 \ 36 \ 2\]

(a) (3 points) Draw the array when 1 element has been sorted using the Bubble Sort algorithm.

Solution: \[8 \ 4 \ 12 \ 20 \ 32 \ 2 \ 36\]

(b) (3 points) Draw the array when 1 element has been sorted using the Selection Sort algorithm.

Solution: \[2 \ 8 \ 4 \ 32 \ 20 \ 36 \ 12\]

4. Consider the following code:

```java
public static void method1(int[] array) {
    int temp = array[1];
    array[1] = array[2];
    array[2] = temp;
}

public static void method2(int[] array) {
    array = new int[5];
    for(int i = 0; i < array.length; i++) {
        array[i] = i;
    }
}

public static void method3(int a1, int a2) {
    int temp = a1;
    a1 = a2;
    a2 = temp;
}

public static void main(String[] args) {
    int[] a = {3, 4, 0, 1, 2};
    //method call from following questions here
}
```

(a) (2 points) Draw the array \(a\) if the comment in `main` is replaced with the statement: `method1(a);`

Solution: \[3 \ 0 \ 4 \ 1 \ 2\]

(b) (2 points) Draw the array \(a\) if the comment in `main` is replaced with the statement: `method2(a);`

Solution: \[3 \ 4 \ 0 \ 1 \ 2\]

(c) (2 points) Draw the array \(a\) if the comment in `main` is replaced with the statement: `method3(a[1], a[3]);`
| Solution:  | 3 | 4 | 0 | 1 | 2 |
5. For each of the following questions, give the output of the code. Write your answer in the box provided next to the code. If the code would result in an infinite loop, give the first 3 outputs and then write “infinite loop.” If the code would result in an error for any reason, you can simply write “error.”

(a) (3 points)

```java
for(int i = 1; i <= 5; i+=2) {
    for(int j = 3; j > 1; j--) {
        System.out.print( (i+j) + " ");
    }
    System.out.print("\n");
}
```

Solution:

4 3
6 5
8 7

Scoring: +1 each line of output

(b) (3 points)

```java
int n = 9;
while (n > 0) {
    System.out.println(n);
    if (n % 3 == 0) {
        n--;
        continue;
    } else if (n % 2 == 0) {
        n -= 3;
    } else if (n == 5) {
        break;
    } else {
        n -= 1;
    }
}
```

Solution: 9
8
5

Scoring: 1 pt each output

(c) (2 points)

```java
for(int i = 0; i < 4; i++) {
    int sum = 1;
    sum = sum + i;
    System.out.println(sum);
}
```

Solution: 1
2
3
4

Scoring: .5 for each correct number.
(d) (3 points) Assume the following method is invoked with `leakyPipes(12)`;

```java
public static void leakyPipes(int n) {
    if (n > 0) {
        if (n % 4 == 0) {
            System.out.println("drop " + n);
            leakyPipes(n-3);
        }
        if (n % 3 == 0) {
            System.out.println("drip " + n);
        }
    }
}
```

Solution:
drop 12

drip 9
drip 12

Scoring: +1 each line of output

(e) (2 points)

```java
int i = 1;
while(i < 10) {
    i *= 2;
    System.out.println(i);
}
```

Solution: 2

4
8
16

Scoring: .5pt each output

(f) (4 points) Circle all of the following loops that will give identical output, assuming that they all use the same value for `n`.

A. ```java
   int i = 1;
   while (i < n) {
       System.out.println(i+1);
       i++;
   }
```

B. ```java
   int i = 1;
   while (i < n) {
       System.out.println(i);
       i++;
   }
```

C. ```java
   int i = 0;
   while (i < n) {
       i = i + 1;
       System.out.println(i);
   }
```

D. ```java
   for (int i = 0; i < n; i++) {
       System.out.println(i+1);
   }
```
6. Consider the following program (with the lines numbered for convenient reference).

```java
1: public class Bugs {
2:   public static void main(String args[]) {
3:       Scanner in = new Scanner(System.in);
4:       System.out.print("Enter a positive integer: ");
5:       int limit = in.nextInt();
6:       int evennum = 0;
7:       while (evennum != limit) {
8:           System.out.println(evennum + " is a number under " + limit);
9:           evennum += 2;
10:       }
11:   }
12: }
```

The intent of the program was to print out the even numbers between 0 and the positive number the user entered. However, there is a bug in the program as written and when the user types in 101, it generates the (abbreviated) output shown below:

0 is a number under 101
2 is a number under 101
...
98 is a number under 101
100 is a number under 101
102 is a number under 101
104 is a number under 101
...
(forever)

(a) (2 points) Explain what the error is in the program.

Solution: This program has an infinite loop in it because the termination condition of the loop will never be fulfilled. The `evennum` variable will only ever contain even numbers and the `limit` variable is odd.

(b) (2 points) What line(s) of code would you change to eliminate this error? Change the line(s) to make the program function correctly, assuming the user enters any valid positive integer. You **do not** need to rewrite the entire program; you can just use the included line numbers and rewrite the problematic line(s) of code.

Solution: Need to change the condition in line 7. It should be something like:

```java
while (evennum <= limit)
```
7. For both of the following questions, you will be writing a method named `printLetters`. This method should take a single String as an input parameter and should return a modified String where the letters are separated by the `'-` character. Examples:

    printLetters("Hello") returns H-e-l-l-o
    printLetters("A") returns A
    printLetters("") returns an empty String

(a) (7 points) Write the method `printLetters` using a loop. You may use any type of loop you wish.

Solution:
```
public static String printLetters(String str) {
    String ns = "";
    if(str.length < 2) {
        return str;
    }

    ns = ns + str.charAt(0);
    for(int i = 1; i < str.length(); i++) {
        ns = ns + "-" + str.charAt(i);
    }
    return ns;
}
```

Scoring:
+2 fencepost “first” post +1 empty strings or strings with 1 char handled correctly +1 uses loop correctly (bounds correct, no infinite loop) +1 builds new String correctly by appending “-” and 1 character. (No scoping issues) +1 returns String +1 returns correct String

For common errors, see list after part (b).

(b) (8 points) Write the method `printLetters` as a recursive method. Methods written with loops will receive no credit.

Solution:
```
public static String printLetters(String str) {
    if (str.length() == 0 || str.length() == 1) {
        return str;
    } else {
        return str.charAt(0) + "-" + questions(str.substring(1));
    }
}
```

+1 method header correct (this is for both part a and part b)
+1 base case for empty string present/correct
+1 base case for string of length 1 present/correct
+1 extracts 1 character and appends “-“ to it
+1 contains a recursive call
+1 shortens string correctly (for smaller problem argument to recursive call)
+1 uses result of recursive call correctly
+1 returns correct value in non-base case situations.
Solution: Common Errors on both parts:

- a return statement inside a loop. Something like:

  ```java
  for (int i = 1; i < str.length(); i++) {
    return ns + "-" + str.charAt(i);
  }
  ```

  return ends the method immediately, so only 1 iteration of the loop executes in the code above.

- Not recognizing the fencepost nature of the problem. Many students wrote code which would return a '-' after every letter, and the end result would be something like “H-e-l-l-o-” instead of just “H-e-l-l-o”.

- A problem with the scope of the string variable used to build up the new string. Something like:

  ```java
  for(int i = 0; i < str.length()-1; i++) {
    String ns = ns + str.charAt(i) + "-";
  }
  ```

- Trying to use a String like an array. You cannot index Strings in the same way you can arrays. So trying to do something like str[i] is incorrect.

- println vs. return. Some students are still confusing prints and returns. This method did not need to print anything. However, it did need to return a String value.

- Mistakes which did not result in point deductions but should be recognized all the same:
  - str.length instead of str.length(). Arrays and strings use different notation!
  - Inadvertent character math. A statement like: `ns = str.charAt(i) + '-' + ns;` will result in the character at index i being added to the hyphen character. Java will treat these as integers (because that’s the only way it can add characters) and give you an integer result. This integer value will then be concatenated to the value of ns.
  - char and String misassignments. Statements like: `String ns = str.charAt(0);` are incorrect because of a data type mismatch. This statement tries to assign a character value to a String typed variable.
8. (8 points) Write a method named **collapse** that takes an array of integers as a parameter. The method should **NOT** modify the input array. Instead, it should return a new array. The new array should be formed by adding pairs of elements: the first two elements, then the next two, then the next two, and so forth. If the array contains an odd number of elements, the last (unpaired) element is simply the last element of the new array. Examples:

collapse({4, 3, 2, 1}) should return the array {7, 3}
collapse({1, -3, -15}) should return the array {-2, -15}
collapse({}) should return the array {}
collapse({13}) should return the array {13}

**Solution:** Answers vary, but here is a very concise answer:

```java
public static int[] collapse(int[] a) {
    // make array of correct length for even or odd inputs
    int len = a.length/2 + a.length%2;
    int[] na = new int[len];

    // populate array regardless of even/odd
    for (int i = 0; i < a.length-1; i+=2) {
        na[i/2] = a[i] + a[i+1];
    }

    // if odd length fix last element
    if (a.length%2 == 1) {
        na[na.length-1] = a[a.length-1];
    }
    return na;
}
```

Here is a slightly longer, but more understandable version:

```java
public static int[] collapse(int[] a) {
    int[] na;
    if (a.length % 2 == 0) { // even length array
        na = new int[a.length/2];
        int idx = 0;
        for (int i = 0; i < a.length; i+=2) {
            na[idx] = a[i] + a[i+1];
            idx++;
        }
    } else { // odd length array
        na = new int[a.length/2 + 1];
        int idx = 0;
        for (int i = 0; i < a.length-1; i+=2) {
            na[idx] = a[i] + a[i+1];
            idx++;
        }
        na[na.length-1] = a[a.length-1]; // deal with last element
    }
    return na;
}
```
function myMethod(a) {
    return na;
}

**Scoring:**
+1 method header correct
+1 creates new array; doesn’t modify input array
+2 correctly populates new array from array w/ even number of elements
+2 correctly populates new array from array w/ odd number of elements
+1 returns an array
+1 returns correct values in array

**Common mistakes:**

- Scope of new array for storing sums. If you declare the variable inside an if or loop, the scope will be limited and cause errors. For example:

```java
for(int i = 0; i < a.length; i+=2) {
    int[] na = new int[a.length/2];
    
    ...
}
```

- Errors with the pattern of sums. Many students tried to use the same variable for indexing into both arrays. This can’t work because the arrays have different lengths. For example:

```java
int[] na = new int[a.length/2];
for(int i = 0; i < a.length; i++) {
    na[i] = a[i] + a[i+1];
}
```

This loop will yield the pattern:

- na[0] = a[0] + a[1]

...  
- na[a.length-1] = runtime error; index out of bounds

Moreover, this stores incorrect values because it doesn’t sum the pairs correctly.

- Not accounting for last element and different size of array which results if input array is odd length.

- Trying to use a nested loop. Many students wrote code like:

```java
for(int i = 0; i < na.length; i++) {
    for (int j = 0; j < a.length-1; j++) {
        na[i] = a[j] + a[j+1];
    }
}
```

Using a nested loop like this will cause every element of na to be set to the sum of the last two elements of a because the assignment statement executes for all values j to a.length-2 (inclusive) every time the inner loop runs.
• Returning an alias instead of a new array. Many students did something like:

    if (a.length == 0 || a.length == 1) {
        return a;
    }

Technically, this returns an alias to a, and not a new array. Code that’s constructed well doesn’t need to deal with these cases directly as they are accounted for automatically (see the sample solutions above and note how they work with input arrays of length 0 or 1 in addition to other arrays). However, I didn’t take points off for this error.
### Excerpt of ASCII Chart

<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>032</td>
<td>Space</td>
</tr>
<tr>
<td>033</td>
<td>!</td>
</tr>
<tr>
<td>034</td>
<td>&quot;</td>
</tr>
<tr>
<td>035</td>
<td>#</td>
</tr>
<tr>
<td>036</td>
<td>$</td>
</tr>
<tr>
<td>037</td>
<td>%</td>
</tr>
<tr>
<td>038</td>
<td>&amp;</td>
</tr>
<tr>
<td>039</td>
<td>'</td>
</tr>
<tr>
<td>040</td>
<td>(</td>
</tr>
<tr>
<td>041</td>
<td>)</td>
</tr>
<tr>
<td>042</td>
<td>*</td>
</tr>
<tr>
<td>043</td>
<td>+</td>
</tr>
<tr>
<td>044</td>
<td>,</td>
</tr>
<tr>
<td>045</td>
<td>-</td>
</tr>
<tr>
<td>046</td>
<td>.</td>
</tr>
<tr>
<td>047</td>
<td>/</td>
</tr>
<tr>
<td>048</td>
<td>0</td>
</tr>
<tr>
<td>049</td>
<td>1</td>
</tr>
<tr>
<td>050</td>
<td>2</td>
</tr>
<tr>
<td>051</td>
<td>3</td>
</tr>
<tr>
<td>052</td>
<td>4</td>
</tr>
<tr>
<td>053</td>
<td>5</td>
</tr>
<tr>
<td>054</td>
<td>6</td>
</tr>
<tr>
<td>055</td>
<td>7</td>
</tr>
<tr>
<td>056</td>
<td>8</td>
</tr>
<tr>
<td>057</td>
<td>9</td>
</tr>
<tr>
<td>058</td>
<td>:</td>
</tr>
<tr>
<td>059</td>
<td>;</td>
</tr>
<tr>
<td>060</td>
<td>&lt;</td>
</tr>
<tr>
<td>061</td>
<td>=</td>
</tr>
<tr>
<td>062</td>
<td>&gt;</td>
</tr>
<tr>
<td>063</td>
<td>?</td>
</tr>
<tr>
<td>064</td>
<td>@</td>
</tr>
<tr>
<td>065</td>
<td>A</td>
</tr>
<tr>
<td>066</td>
<td>B</td>
</tr>
<tr>
<td>067</td>
<td>C</td>
</tr>
<tr>
<td>068</td>
<td>D</td>
</tr>
<tr>
<td>069</td>
<td>E</td>
</tr>
<tr>
<td>070</td>
<td>F</td>
</tr>
<tr>
<td>071</td>
<td>G</td>
</tr>
<tr>
<td>072</td>
<td>H</td>
</tr>
<tr>
<td>073</td>
<td>I</td>
</tr>
<tr>
<td>074</td>
<td>J</td>
</tr>
<tr>
<td>075</td>
<td>K</td>
</tr>
<tr>
<td>076</td>
<td>L</td>
</tr>
<tr>
<td>077</td>
<td>M</td>
</tr>
<tr>
<td>078</td>
<td>N</td>
</tr>
<tr>
<td>079</td>
<td>O</td>
</tr>
<tr>
<td>080</td>
<td>P</td>
</tr>
<tr>
<td>081</td>
<td>Q</td>
</tr>
<tr>
<td>082</td>
<td>R</td>
</tr>
<tr>
<td>083</td>
<td>S</td>
</tr>
<tr>
<td>084</td>
<td>T</td>
</tr>
<tr>
<td>085</td>
<td>U</td>
</tr>
<tr>
<td>086</td>
<td>V</td>
</tr>
<tr>
<td>087</td>
<td>W</td>
</tr>
<tr>
<td>088</td>
<td>X</td>
</tr>
<tr>
<td>089</td>
<td>Y</td>
</tr>
<tr>
<td>090</td>
<td>Z</td>
</tr>
<tr>
<td>091</td>
<td>[</td>
</tr>
<tr>
<td>092</td>
<td>\</td>
</tr>
<tr>
<td>093</td>
<td>]</td>
</tr>
<tr>
<td>094</td>
<td>^</td>
</tr>
<tr>
<td>095</td>
<td>_</td>
</tr>
<tr>
<td>096</td>
<td>'</td>
</tr>
<tr>
<td>097</td>
<td>a</td>
</tr>
<tr>
<td>098</td>
<td>b</td>
</tr>
<tr>
<td>099</td>
<td>c</td>
</tr>
<tr>
<td>100</td>
<td>d</td>
</tr>
<tr>
<td>101</td>
<td>e</td>
</tr>
<tr>
<td>102</td>
<td>f</td>
</tr>
<tr>
<td>103</td>
<td>g</td>
</tr>
<tr>
<td>104</td>
<td>h</td>
</tr>
<tr>
<td>105</td>
<td>i</td>
</tr>
<tr>
<td>106</td>
<td>j</td>
</tr>
<tr>
<td>107</td>
<td>k</td>
</tr>
<tr>
<td>108</td>
<td>l</td>
</tr>
<tr>
<td>109</td>
<td>m</td>
</tr>
<tr>
<td>110</td>
<td>n</td>
</tr>
<tr>
<td>111</td>
<td>o</td>
</tr>
<tr>
<td>112</td>
<td>p</td>
</tr>
<tr>
<td>113</td>
<td>q</td>
</tr>
<tr>
<td>114</td>
<td>r</td>
</tr>
<tr>
<td>115</td>
<td>s</td>
</tr>
<tr>
<td>116</td>
<td>t</td>
</tr>
<tr>
<td>117</td>
<td>u</td>
</tr>
<tr>
<td>118</td>
<td>v</td>
</tr>
<tr>
<td>119</td>
<td>w</td>
</tr>
<tr>
<td>120</td>
<td>x</td>
</tr>
<tr>
<td>121</td>
<td>y</td>
</tr>
<tr>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>123</td>
<td>{</td>
</tr>
<tr>
<td>124</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>}</td>
</tr>
<tr>
<td>126</td>
<td>~</td>
</tr>
<tr>
<td>127</td>
<td>DEL</td>
</tr>
</tbody>
</table>
THIS PAGE LEFT INTENTIONALLY BLANK