# **CIS 36A :: LAB 8 - Recursions**

#### **Student Name: Esmatullah Nickzad**

### **Task 1: Definitions & Concepts**

**Instructions: Briefly** answer the questions below.

Exercise

### **Task 2: Understanding Programming**

Instructions: Answer each question below.

Exercise 5: Write a **recursive** method that displays the content of a string backward.

|  |
| --- |
| Answer: Public static void displayBackward (String text) {  If (text.length() ==0) {  Return;  } else {  System.out.print(text.charAt(text.length()( -1));  displayBackward(text.substring(0, text.length() -1)); }  } |

Exercise 25: The **drawStars( )** method described in the section on recursion draws one star and then recursively calls itself to draw the remaining **n**-1 stars. Could it also do things in the opposite order? That is, could it recursively call itself to draw **n**-1 of the stars and then draw the last star?

|  |
| --- |
| **Answer:**  public class DrawStars {  public static void drawStars(int n) {  if (n == 0) {  return;  }  drawStars(n - 1);  System.out.print("\* ");  }  public static void main(String[] args) {  int numStars = 5;  drawStars(numStars);  }  } **we can also do it in other way:**  public static void drawStars(int n) {  if (n > 0) {  drawStars(n - 1);  System.out.print("\*");  }  } |

Exercise 26: What would happen if the drawStars( ) method described in the section on recursion were invoked using as its argument the integer -1? Modify the method so that nothing is printed if a negative integer is passed in as its argument.

|  |
| --- |
| **Answer:**  public static void drawStars(int n) {  if (n >= 0) {  if (n > 0) {  drawStars(n - 1); /  System.out.print("\*");  } else {  // Print nothing for n = 0 (optional, base case already handles this)  }  } else {  // Do nothing for negative input to avoid infinite recursion  }  }  In the above I have , I have written IF statement to check for non-negative input, and also draw the ramaining n-1 stars recursively , also draw the last star , and print nothing for n=0. |

### **Task 3: Programming Exercises**

Instructions: Use any IDE to write and execute below exercises from the book chapter 3. Attach Snipping photos of your source code and execution of the code in the console. Make sure to create separate files for each exercise.

**Chapter Exercises: Do the following chapter exercises.**

Exercise 27: Write a recursive method countDown( ) that takes an integer n as its parameter. It prints the integers from n down to 0, one per line, and then prints"Blast off!".

A screen shot of a computer program

Description automatically generated

Exercise 28: Write a recursive method **add1toN( )** that takes an integer **n** as its parameter. It returns the sum 1 + 2 + 3 + ... + **n**.

A screen shot of a computer program

Description automatically generated

Exercise 30 - Implement an **equalArrays( )** method that takes two integer arrays as parameters and returns **true** if both arrays are the same length and have equal values at corresponding indices.

B: Recursively (Hint: Create a helper method with an extra parameter)

A screenshot of a computer program

Description automatically generated

Exercise 31 - A: Write a method **reverse( )** that takes an integer array as its parameter and reverses the order of the elements in the array.

B: Recursively (Hint: Create a helper method with an extra parameter)

A screen shot of a computer program

Description automatically generated

Exercise 32: Implement a numTimes( ) method that takes two parameters: an integer array called data and an integer called x. It returns the number of times that x appears in the array.

B: Recursively (*Hint:* Create an auxiliary function with an extra parameter.)

A screen shot of a computer program

Description automatically generated

**Task 4 - Programming Application**

A rational number is a number that can be represented as the ratio of two integers. For example, 2/3 is a reasonable number, and you can think of 7 as a rational number with an implicit 1 in the denominator or 7/1.

**Continue from Lab 7:**

1. Write a void **recursive** method named **reduce()** that reduces a rational number to its lowest terms by finding the greatest common divisor (GCD) of the numerator and denominator and dividing through. This method should be a mutator and should modify the instance variables of the object on which it is invoked.  
   *Hint:* Finding the GCD only takes a few lines of code. Search the web for “Euclidean algorithm”.

A screenshot of a computer program

Description automatically generated