Kyler Grahame
CSIT 313
Homework #4

1. (3 pts) Java supports multiple selection statements, for example, using if else if else. #IF-ELSE-STATEMENTS

a) If the first else (red else) was not included in the statement, to which if (first or second) is the remaining else associated?

i. Create and execute a Java program that illustrates your responses. Title this program Task1a.java

ii. Answer in a document the question and explain how your code supports your statements

Answer 1a:

- The else is associated with the second if. We can tell this by the output being (num = 5) and then (num = 30). If it was associated with the first if then num = 30 would not print at all. So the code goes through this flow. if (num == 10) then num = 5 (reassigns value) and then it prints num = 5. Then it moves to the second if and (num == 10) but now since num was reassigned it does not != 10 so it moves to the else statement where num = 30 and prints num is 30. (Look at code Below)

```
// Task1A
    public class Task1A
    {
        public static void main(String[] args)
        {
            int num = 10;
            if (num == 10) // 1st if
            {
                num = 5;
                System.out.println("Num is = " + num);
            }
            // 1st else removed
            if (num == 10) //2nd if
            {
                num = 15;
```

```
                System.out.println("Num is = " + num);
            }
            else //2nd else
            {
                num = 30;
                System.out.println("Num is = " + num);
            }
        }
    }
```

b) Describe how would you modify the construct so the second else is associated with the other if?

i. Create and execute a Java program that illustrates your responses. Title this program Task1b.java

ii. Answer in a document the question and explain how your code supports your statements

Answer 1b:

- So to associate the second else with the first if you can nest the second if else block inside the first else block. This was if (num == 10) is not conditionally reached it will activate the second if and else conditions of the second if else block. So the printed results are num = 5. because in the first if block it is if (num == 10) then num is reassigned to 5 and num is = 5 is printed and then it closes the program because the else won't be reached with the if else nested block.

```
// Task1B
    public class Task1B
    {
        public static void main(String[] args)
        {
            int num = 10;
            if (num == 10) // 1st if
            {
                num = 5;
                System.out.println("Num is = " + num);
            }
            else //1st if and nest the second if else block
            {
                if (num == 10) //2nd if
                {
```

```
                    num = 15;
                    System.out.println("Num is = " + num);
                }
                else //2nd else
                {
                    num = 30;
                    System.out.println("Num is = " + num);
                }
            }
        }
    }
```

2. (2 pts) In short-circuit evaluation of an expression the result is determined without evaluating all of the operands and/or operators. Consider an if (Boolean expression) else selection statement in Java. #ShortCircuitEvaluation

a) Does Java support short-circuit evaluation of the Boolean expression? Add to the Java evaluating all of the operands and/or operators. Consider an if (Boolean expression) else selection statement in Java.

```
i. Create and execute a Java program that illustrates your responses. Title
this program Task2a.java

ii. Answer in a document the question and explain how your code supports your
statements
```

Answer 2A:

- Java does support short circuit evaluation. In my code I use the AND operator && over the or operator. So when the if statement goes if (num == 10 && isBoolTrue()) since num = 5, the first condition is met and the isBoolTrue() function is not executed. This shows that Java supports short circuit evaluation as it does not execute the function as the function is irrelevant in this context so should not be executed for the sake of efficiency.

```
// Task2A
public class Task2A
{
    public static void main(String[] args)
    {
        int num = 9;
        // Short-circuit evalution
        if (num == 10 && isBoolTrue() )
```

```
        {
            System.out.println("Both conditions = True");
        }
        else
        {
            System.out.println("Short circuit evaluation occured, second
condition not evaluated.");
        }
    }

    public static boolean isBoolTrue()
    {
        System.out.println("Second condition met");
        return false;
    }
}
```

b) If Java does not support short-circuit evaluation, add a code segment which enforces short-circuit evaluation. If Java does support short-circuit evaluation, add a code segment that enforces full evaluation.

```
i. Create and execute a Java program that illustrates your responses. Title
this program Task2b.java

ii. Answer in a document the question and explain how your code supports your
statements
```

Answer for 2b:

- The first if statement checks if (num == 10), if it is equivalent to 10 then it will proceed to the next if else block that is nested inside. If it is not equivalent it will move to the connected else statement and will have no conditions met. This behaves identical to the short circuit with operands and is a full evaluation as you can see all the output regarding the process of the execution flow.

```
//Task 2BS
public class Task2A
{
    public static void main (String[] args)
    {
            int num = 9;
            if (num == 10)
            {
```

```
                if (isBoolTrue())
                { // This only evaluates if num == 10 is true
                    System.out.println("Both conditions = True");
                }
                else
                {
                    System.out.println("Second condition was false");
                }
            }
            else
            {

                System.out.println("First condition was false, second
condition not evaluated.");
            }
        }
}

    public static boolean isBoolTrue()
    {
        System.out.println("Second condition met");
        return false;
    }
}
```

3. (2 pts) Problem 5 (Programming Exercises Chapter 8, page 363) from the textbook (the question with go to reject) #CODE_READABILITY

a. Create and execute a Java program that illustrates your responses. Title this program Task3.java

b. Create and execute a program in a language different from Java that illustrates your responses. Title this program Task3...

c. Answer in a document the problem and explain how your code supports your statements. In addition, compare the two languages

Code for 3A:

```
public class Task3
{
    public static void main(String[] args)
    {
        int n = 3; // EX: Matrix size
        int[][] x =
```

```
        {               // INDEX of rows below:
            {1, 0, 0}, // row 0
            {0, 0, 0}, // row 1
            {0, 0, 0}  // row 2
        };

            boolean allZero = false; // flag for checking for all zero
            for (int i = 0; i < n; i++)
            {
                allZero = true; //We can assume a row is all-zero unntil
discovered otherwise
                for (int j = 0; j < n; j++)
                {
                    if(x[i][j] != 0)
                    {
                        allZero = false; // if row has non-zero value flag =
false
                        break; //exit
                    }
                }
                if (allZero)
                {
                    System.out.println("First all zero is: " + i);
                    break; //exit for first loop if row is all zero
                }
            }
            if(!allZero)
            {
                System.out.println("No all zero rows.");
            }
    }
}
```

Console Output for 3A:

```
Listening on 57204
User program running
User program finished
First all zero is: 1
```

Code for 3B:

```
def Task3(matrix):
    n = len(matrix) # Dynamic initialize Matrix Size
```

```
    allZero = False

    for i in range(n):
        allZero = True
        for j in range(n):
            if matrix[i][j] != 0:
                allZero = False
                break # exit second loop
        if allZero:
            print("First all zero is: ", i)
            break
    if not allZero:
        print("No all zero rows.")
#EX: matrix
x = [
    [0,1,0], # Row 0
    [1,0,1], # Row 1
    [0,0,0]  # Row 2
]
Task3(x) # Runs the function and returns either i = 0, 1, or 2 based off the
rows of the matrix from top to bottom. Will print the row index correlating to

         # which row has all zero and is checked first.
```

Console Output for 3B:

```
First all zero is:  2
```

Answer for 3:

- In comparison to the goto code provided in the book I think using bools as flags is much more readable. Flags in general are very easily understandable as they have 2 or possible 3 states of either True, False, and null so they are simple. They also if named properly are very easy to seen in hardcode through either reassignment or variable declarement and initialization, for example: bool isThreadReady = true; As you can see in that example that is Extremely readable as well as straight forward. Also slightly unrelated but in my experience taking your time to break up brackets properly and use white space indentation if allowed makes code much prettier, less cluttered, and more readable by far as I have done this in my java code in comparison to the provided code for goto. the python code in this example is a little more difficult to follow but only due to python's dynamic runtime compiling because it makes the python syntax less clear in examples such as in a java for

loop you can clearly see all of the preconditions for the loops lifetime but as in Python its a less concise using range(n) at times.