CIS 36B – 3rd Class / Lab Assignment – **10 Points –**
Student Name Kachilau **Student ID** 10819338 **Point Total**

Section 1: Define / Answer

Logical Operators

& - bitwise and

| - bitwise or

^ - bitwise xor

|| -logical or

&& - logical and

! - not

Difference between & and &&?

`&` is bitwise. `&&` is logical.
`&` evaluates both sides of the operation.
`&&` evaluates the left side of the operation, if it's `true`, it continues and evaluates the right side.

Complete Logic Table

P	S	P &S	P | S	P ^ S	!S	(!(P^S)) &(P|S)
TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE

(P&S) && (P|S)	(P^S) || (!P)
FALSE	TRUE

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name Kachilau **Student ID** 10819338 **Point Total**

FALSE	**FALSE**
TRUE	**TRUE**
FALSE	**FALSE**

Data Structure:

https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

1. LinkedList-an ordered set of data elements, each containing a link to its successor (and sometimes its predecessor).

Describe Differences between LinkedList and ArrayList-

LinkedList and ArrayList both implement List Interface but how they work internally is where the differences lies.
Main **difference between ArrayList and LinkedLis**t is that ArrayList is implemented using re sizable array while LinkedList is implemented using doubly LinkedList. ArrayList is more popular among Java programmer than LinkedList as there are few scenarios on which LinkedList is a suitable collection than ArrayList.

Pg. 577, Java Programming *A comprehensive Introduction*

http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format(Detailed explanation of Java documentation)

http://www.tutorialspoint.com/java/java_documentation.htm

http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments

Internal Documentation- the notes on how and why various parts of code operate is included within the source code as comments. It is often combined with meaningful variable names with the intention of providing potential future programmers a means of understanding the workings of the code.

Internal documentation would be comments and remarks made by the programmer in the form of line comments and boiler plates.

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name Kachilau **Student ID** 10819338 **Point Total**

External Documentation- External documentation would be things like flow charts, UML diagrams, requirements documents, design documents etc.

Java Doc Tags- is a documentation generator from Oracle Corporation for generating APIdocumentation in HTML format from Java source code. The HTML format is used to add the convenience of being able to hyperlink related documents together.[2]

Javadoc tags (Examples)

Tag	Description	Syntax
@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecated-text
@exception	Adds a **Throws** subheading to the generated documentation, with the class-name and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. T	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name Kachilau **Student ID** 10819338 **Point Total**

@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant:	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name Kachilau **Student ID** 10819338 **Point Total**

Programming Assignment

Task 1- (Using the &&, || and ^ operators) Write a program that prompts the user to enter an integer and determines whether it is divisible by 5 or 6, whether by 5 or 6, and whether it is divisible by 5 or 6, but not both.

Sample Output

Enter an Integer: 10

Is 10 divisible by 5 and 6? false

Is 10 divisible by 5 or 6? false

Is 10 divisible by 5 or 6, but not both? true

CIS 36B – 3rd Class / Lab Assignment – **10 Points** –

Student Name Kachilau **Student ID** 10819338 **Point Total**

```
1
2   package javaapplication1;
3
5   import java.io.*;
    import java.util.Scanner;
    import java.util.*;
7
8   public class JavaApplication3 {
9
10      public static void main(String[] args) {
11
12          Scanner input = new Scanner(System.in);
13
14          System.out.print("Please enter a integer: ");
15          int x = input.nextInt();
16
17          System.out.print("Is " + x + " divisible by 5 and 6?");
18          if(((x % 5) == 0) && ((x % 6) ==  0)) {
19              System.out.println(" true");
20          } else {
21              System.out.println(" false");
22          }
23
24          System.out.print("Is " + x + " divisible by 5 or 6?");
25          if(((x % 5)== 0) && ((x % 6)== 0)) {
26              System.out.println(" true");
27          } else {
28              System.out.println(" false");
29          }
30
31          System.out.print("Is " + x + " divisible by 5 or 6, but not both?");
32          if(((x % 5)== 0) || ((x % 6)== 0)) {
33              System.out.println(" true");
34          } else {
35              System.out.println(" false");
36          }
37      }
38  }
39
```

JavaApplication3 > main >

Output - JavaApplication1 (run)

```
run:
Please enter a integer: 10
Is 10 divisible by 5 and 6? false
Is 10 divisible by 5 or 6? false
Is 10 divisible by 5 or 6, but not both? true
BUILD SUCCESSFUL (total time: 2 seconds)
```

CIS 36B – 3rd Class / Lab Assignment – **10 Points** –

Student Name Kachilau **Student ID** 10819338 **Point Total**

Task 2- Create a program that uses a **switch** statement and **case** statements to output the number of days in any given month based upon a users input.

February is a unique month. So the **case** for February will need a **if**… **else** to capture the two possible options for the number of days in February based upon leap year. Include an ability to input the year if the user chooses February.

If February is a leap year you will need logical operators and circuit operators to print the number of days in a leap year.

Sample Output

Enter a month: 6

There are 30 Days in June

Enter a month: 2

Enter a year: 2012

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name **Kachilau** **Student ID** **10819338** **Point Total**

There 29 Days in February

CIS 36B – 3rd Class / Lab Assignment – **10 Points** –

Student Name Kachilau **Student ID** 10819338 **Point Total**

```java
package javaapplication1;

import java.util.Scanner;

public class JavaApplication5 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a month: ");
        int month = input.nextInt();

        switch (month) {
            case 0:
                System.out.println("Invalid month");
                break;
            case 1:
                System.out.println("There are 31 Days in Januray");
                break;
            case 2:
                System.out.print("Enter a year: ");
                int year = input.nextInt();

                boolean leapyear = ((year % 4 == 0) && (year % 100 != 0) || year % 400 == 0);

                if(leapyear) {
                    System.out.println("There are 29 Days in Februnary");
                } else {
                    System.out.println("There are 28 Days in Februnary");
                }
                break;
            case 3:
                System.out.println("There are 31 Days in March");
                break;
            case 4:
                System.out.println("There are 30 Days in April");
                break;
            case 5:
                System.out.println("There are 31 Days in May");
                break;
            case 6:
                System.out.println("There are 30 Days in June");
                break;
            case 7:
                System.out.println("There are 31 Days in July");
                break;
            case 8:
                System.out.println("There are 31 Days in August");
                break;
            case 9:
                System.out.println("There are 30 Days in September");
                break;
            case 10:
                System.out.println("There are 31 Days in October");
                break;
            case 11:
                System.out.println("There are 30 Days in November");
                break;
            case 12:
                System.out.println("There are 31 Days in December");
                break;
            default:
```

CIS 36B – 3rd Class / Lab Assignment – **10 Points** –

Student Name Kachilau **Student ID** 10819338 **Point Total**

```
62                  System.out.println("Invalid month");
63
64              }
65
66          }
67
68      }
69
```

JavaApplication5 > main > switch (month) >

Output - JavaApplication1 (run)

```
run:
Enter a month: 2
Enter a year: 2012
There are 29 Days in Februnary
BUILD SUCCESSFUL (total time: 8 seconds)
```

Task 3-

Math.random() is a method in the Java library that computes a random **double** value between 0 and 1.

For example:

double x = Math.random();

assigns to the variable x a random **double** between 0 and 1. Write a program that tests how well **Math.randon()** works. Write a program that calls **Math.random()** 1,000 times to create 1,000

values. Store the values that are greater than 0.3 and .7 in a LinkedList and then print all the elements in the LinkedList. Store values other values in an ArrayList and then print all the values.

```
1
2   package javaapplication1;
3
4   import java.util.*;
5
6   public class JavaApplication2 {
7
8       public static void main(String[] args) {
9           LinkedList<String> l = new LinkedList<String>();
10          ArrayList<String> a = new ArrayList<String>();
11
12          double x = 0;
13
14          for(int i = 0; i <= 1000; i++){
15              x = Math.random();
16
17              if((x > 0.3) && (x < 0.7)){
18                  l.add(Double.toString(x));
19              } else {
20                  a.add(Double.toString(x));
21              }
22          }
23
24          System.out.println("Integers between 0.3 and 0.7: " + l);
25          System.out.println("Integers not between 0.3 and 0.7: " + a);
26
27      }
28
29  }
30
```

JavaApplication2 > main

Output - JavaApplication1 (run)

```
run:
Integers between 0.3 and 0.7: [0.5953652961690322, 0.6904095090476962, 0.4580000232739343, 0.4116181876812163,
Integers not between 0.3 and 0.7: [0.019617933065537385, 0.1390092539502621, 0.27306626810860524, 0.99096802538
BUILD SUCCESSFUL (total time: 0 seconds)
```

CIS 36B – 3rd Class / Lab Assignment – **10 Points –**

Student Name **Kachilau** **Student ID** **10819338** **Point Total**