CS1083
Assignment #6

Daniyal Khan
3765942

File1.java

```
import java.io.*;

public class file1 {
    public static void main (String[] args) {
        try {
            FileInputStream f1 = new
FileInputStream("File1.bin"); // reads raw byte data from file
(serialized)
            ObjectInputStream inStream = new
ObjectInputStream(f1);  // deserializes object or primitive data

            while (true) {
                try {
                    char c = inStream.readChar(); // reads
enough data to make a char
                    System.out.print(c);
                }
                catch (EOFException eofe) {
                    break;
                }
            }
            inStream.close();
        } catch (FileNotFoundException fnfe) {
            System.out.println(fnfe.getMessage());
        } catch (IOException ioe) {
            System.out.println(ioe.getMessage());
        }
    }
}
```

File2.java:

```
import java.io.*;

public class file2 {
    public static void main (String[] args) {
        try {
            FileInputStream f2 = new
FileInputStream("File2.bin");
            ObjectInputStream inStream = new
ObjectInputStream(f2);

            while (true) {
                try {
                    char c = inStream.readChar();
                    if (c == 'G') {
                        c = '7';
                    }
                    System.out.print(c);
                } catch (EOFException eofe) {
                    break;
                }
            }
            inStream.close();
        } catch (FileNotFoundException fnfe) {
            System.out.println(fnfe.getMessage());
        } catch (IOException ioe) {
            System.out.println(ioe.getMessage());
        }
    }
}
```

File3.java:

```
import java.io.*;

public class file3 {
    public static void main(String[] args) {
        try {
            FileInputStream f3 = new
FileInputStream("File3.bin");
            ObjectInputStream inStream = new
ObjectInputStream(f3);

            double latitude = inStream.readDouble();
            double longitude = inStream.readDouble();

            System.out.println("Longitude: " + latitude);
            System.out.println("Latitude: " + longitude);

            inStream.close();
        } catch (FileNotFoundException fnfe) {
            System.out.println(fnfe.getMessage());
        } catch (IOException ioe) {
            System.out.println(ioe.getMessage());
        }
    }
}
```

Result:

```
The meeting will be on 10:00am on Thursday at
Longitude: 45.950102
Latitude: -66.642087
```

Part 2:

Iteration.java:

```
import java.util.Scanner;

public class Iteration {
    public static void main (String[] args) {
        try {
            Scanner scan = new Scanner(System.in);
        System.out.println("Enter a integer: ");
        int userInput = Integer.parseInt(scan.nextLine());

        long startTime = System.nanoTime();
        // System.out.println(loop(userInput));
        System.out.println(Recursion(userInput));

        double elapsedSeconds = (double) (System.nanoTime() -
startTime) / 1000_000_000; // converting from nanoseconds to
seconds
        System.out.println("Runtime:" + elapsedSeconds);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public static int loop(int userInput) {
        if (userInput <= 3) {
            return userInput;
        }
        int cs1 = 1;
        int cs2 = 2;
        int cs3 = 3;
        int csn = 0;

        for (int i = 4; i <= userInput; i++) {
            csn = cs3 + cs2 - cs1 + 1;
            cs1 = cs2;
            cs2 = cs3;
            cs3 = csn;
        }
        return csn;
    }

    /* Different approach
    public static int loop2(int n) {
        int CS[] = new int[n+1];
        CS[3] = 3;
        CS[2] = 2;
        CS[1] = 1;

        for (int i = 4; i <= n; i++) {
            CS[i] = CS[i-1] + CS[i-2] - CS[i-3] + 1;
        }
        return CS[n];
    }
    */

    public static int Recursion(int n) {
        if (n <= 3) {
            return n;
        }
        return Recursion(n-1) + Recursion(n-2) - Recursion(n-3)
+ 1;
    }
}
```

	Iterative	Recursive
n=10	2.01708E-4	2.015E-4
n=20	1.98375E-4	0.001743084
n=30	1.575E-4	0.050803833
n=40	1.84209E-4	8.247968458
n=50000	0.002337042	StackOverFlowError

1. With smaller values of n the time difference between the two approaches is negligible.

2. As the value of n increases the time difference increases between the two with Recursive approach being slower. This happens because of all the recursive calls that take up memory in the stack.