Software testing , validation and verification (CSE 338)

Lab Task 2

Submitted to:

Dr. Islam Ahmed Mahmoud elmaddah

Engineer Omar talaat

Made By:

Anas Salah

19P9033

Group 1 Section 1

Question 1 :

Code to check for Even and Odd numbers :

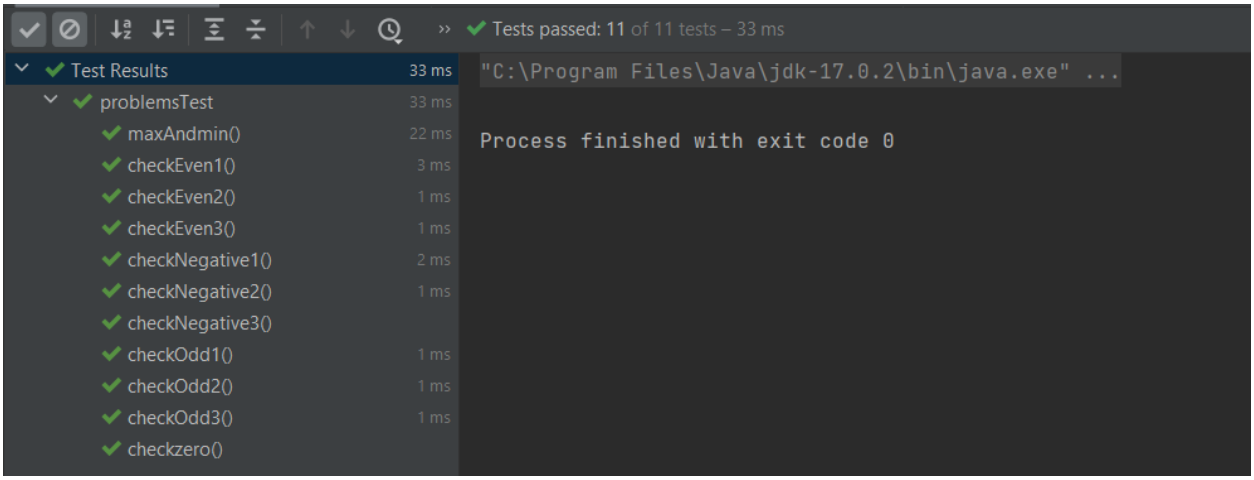
```
/* This function returns "even" if passed number is even and "odd" if passed
number is odd*/

public String EvenOddChecker(int n) {
    if (n<0){
        throw new IllegalArgumentException("Unaccepted Number");
    }
    if (n % 2 == 0)
        return "even";
    else
        return "odd";

}
```

Test cases:

```
/* Tests for EvenOddChecker */
@Test
void checkEven1() {
    problems tester = new problems();
    assertEquals("even",tester.EvenOddChecker(44));
}
@Test
void checkEven2() {
    problems tester = new problems();
    assertEquals("even",tester.EvenOddChecker(2));

}
@Test
void checkEven3() {
    problems tester = new problems();
    assertEquals("even",tester.EvenOddChecker(1231432));
}
@Test
void checkzero() {
    problems tester = new problems();
    assertEquals("even",tester.EvenOddChecker(0));
}
@Test
void checkOdd1() {
    problems tester = new problems();
    assertEquals("odd",tester.EvenOddChecker(31));
}
@Test
void checkOdd2() {
    problems tester = new problems();
    assertEquals("odd",tester.EvenOddChecker(1));
}
@Test
```

```java
void checkOdd3() {
    problems tester = new problems();
    assertEquals("odd",tester.EvenOddChecker(23451));
}
@Test
void checkNegative1() {
    problems tester = new problems();
    assertThrows(IllegalArgumentException.class,() -> {
        tester.EvenOddChecker(-1);
    });
}
@Test
void checkNegative2() {
    problems tester = new problems();
    assertThrows(IllegalArgumentException.class,() -> {
        tester.EvenOddChecker(-14);
    });
}
@Test
void checkNegative3() {
    problems tester = new problems();
    assertThrows(IllegalArgumentException.class,() -> {
        tester.EvenOddChecker(-142141);
    });
}
```

Test case result:



Code for finding max and min element in array:

```
/*This function return an array,
 This array contains the maximum in its first index and the minimum in its
second index*/

public int[] maxAndmin(int[] numbers) {
    int[] minAndmaxArr = new int[2];
    if (numbers.length == 0){
        throw new IllegalArgumentException("empty array");
    }
    int maxValue = numbers[0];
    for(int i=1;i < numbers.length;i++){
        if(numbers[i] > maxValue){
            maxValue = numbers[i];
        }
    }
    minAndmaxArr[0]=maxValue;
    int minValue = numbers[0];
    for(int i=1;i<numbers.length;i++){
        if(numbers[i] < minValue){
            minValue = numbers[i];
        }
    }
    minAndmaxArr[1]=minValue;
    return minAndmaxArr;

}
```

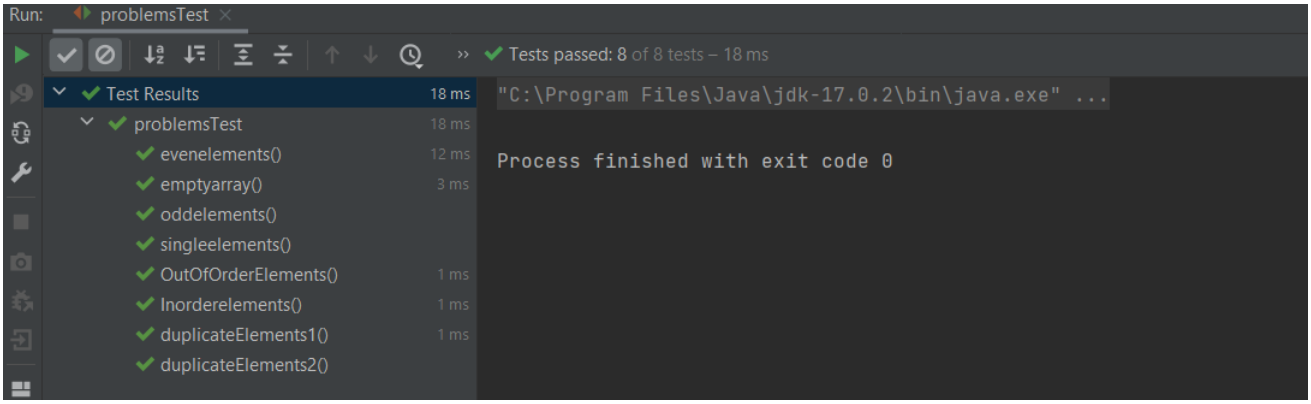
Test cases:

```
/* Tests for maxAndmin */
@Test
void evenelements() {
    problems tester = new problems();
    int[] test={ 12,44,34,3,6,50,33,40 };
    int[] maxmin = {50,3};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);

}

@Test
void oddelements() {
    problems tester = new problems();
    int[] test={ 23,56,63,1,35,123,53 };
    int[] maxmin = {123,1};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
```

```
    assertTrue(result);
}

@Test
void emptyarray() {
    problems tester = new problems();
    int[] test={};
    assertThrows(IllegalArgumentException.class,() -> {
        tester.maxAndmin(test);
    });

}

@Test
void singleelements() {
    problems tester = new problems();
    int[] test={ 23 };
    int[] maxmin = {23,23};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);
}

@Test
void duplicateElements1() {
    problems tester = new problems();
    int[] test={23,16,13,10,15,23,23 };
    int[] maxmin = {23,10};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);
}

@Test
void duplicateElements2() {
    problems tester = new problems();
    int[] test={54,54,54,54,54};
    int[] maxmin = {54,54};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);
}

@Test
void Inorderelements() {
    problems tester = new problems();
    int[] test={1,2,3,4,5,6,7,8};
    int[] maxmin = {8,1};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);
}

@Test
void OutOfOrderElements() {
    problems tester = new problems();
    int[] test={8,7,6,5,4,3,2,1};
```

```
    int[] maxmin = {8,1};
    int[] actual = tester.maxAndmin(test);
    boolean result = Arrays.equals(maxmin,actual);
    assertTrue(result);
}
```

Test cases Result:



Question 3 sheet 3:

Code:

```
/* This funtion takes a string as input containing several button clicks of a
b c d and prints out
    current state , innerstate , date and time
 */
public void question3 (String myinput){
    String state = "Normal Display";
    String innerState = "Time";
    int m=0,h=0, D=1,M=1, Y=2000;
    myinput.toLowerCase();
    for (int i = 0; i < myinput.length(); i++) {
        char currentChar = myinput.charAt(i);
        switch (state){
            case "Normal Display":
                if (currentChar == 'c'){
                    state = "Update";
                    innerState = "min";
```

```
                }
                if (currentChar == 'b'){
                    state = "Alarm";
                    innerState = "Alarm";
                }
                if (currentChar == 'a'){
                    if (innerState == "Time"){
                        innerState = "Date";
                    }
                    else {
                        innerState ="Time";
                    }
                }
                if (currentChar == 'd'){
                    System.out.println("No action in this state with input
d");
                }
                break;
            case "Alarm":
                if (currentChar == 'a'){
                    if (innerState == "Alarm"){
                        innerState = "Chime";
                    }
                }
                if (currentChar == 'b'){
                    System.out.println("No action in this state with input
b");
                }
                if (currentChar == 'c'){
                    System.out.println("No action in this state with input
c");
                }
                if (currentChar == 'd'){
                    state = "Normal Display";
                    innerState = "Time";
                }
                break;
            case "Update":
                if (currentChar == 'a'){
                    switch (innerState){
                        case "min":
                            innerState = "hour";
                            break;
                        case "hour":
                            innerState = "day";
                            break;
                        case "day":
                            innerState = "month";
                            break;
                        case "month":
                            innerState = "year";
                            break;
                        case "year":
                            state = "Normal Display";
                            innerState = "Time";
                            break;
                    }
```

```
                }
                if (currentChar == 'b'){
                    switch (innerState){
                        case "min":
                            m++;
                            if (m == 60){
                                m = 0;
                                h++;
                                if (h==24){
                                    h=0;
                                    D++;
                                    if (D == 31){
                                        D=1;
                                        M++;
                                        if (M==13){
                                            M=1;
                                            Y++;
                                        }
                                    }

                                }
                            }
                        case "hour":
                            h++;
                            if (h==24){
                                h=0;
                                D++;
                                if (D == 31){
                                    D=1;
                                    M++;
                                    if (M==13){
                                        M=1;
                                        Y++;
                                    }
                                }

                            }
                        case "day":
                            D++;
                            if (D == 31){
                                D=1;
                                M++;
                                if (M==13){
                                    M=1;
                                    Y++;
                                }
                            }
                        case "month":
                            M++;
                            if (M==13){
                                M=1;
                                Y++;
                            }
                        case "year":
                            Y++;

                    }
```

```
                }
                if (currentChar == 'c'){
                    System.out.println("No action in this state with input
c");
                }
                if (currentChar == 'd'){
                    state = "Normal Display";
                    innerState = "Time";
                }
                break;

        }
        System.out.println("Current State is : " + state);
        System.out.println("Current innerState is : " + innerState);
        System.out.println("DATE: " + Y + " - " + M + " - " + D);
        System.out.println("TIME: " + h + " : " + m);

    }

}
```

Code adjustment:

- I needed to make a few adjustments to the code to be able to test the function using j unit
- My approach was to write the output to a text file before printing it out
- Then I write my expected output in a different text file with the same format as the output
- Then I used a simple assertequals() in the test cases to compare the output from the function and the expected output

Code after adjustments:

```
/* This funtion takes a string as input containing several button clicks of a
b c d and prints out
        current state , innerstate , date and time
     */
ArrayList<String> output = new ArrayList<String>();
String myInput;

public String getMyInput() {
    return myInput;
}

public void setMyInput(String myInput) {
    this.myInput = myInput;
}

public void State() throws IOException {
    String myInput = getMyInput();
    if (myInput.length() == 0) {
        File f = new File("Output.txt");
        FileOutputStream fos = new FileOutputStream(f);
        PrintWriter printwrite = new PrintWriter(fos);
        output.add("Your Input is empty");
        printwrite.write(String.valueOf(output));
        printwrite.flush();
        fos.close();
        printwrite.close();
    } else {
        File f = new File("Output.txt");
        FileOutputStream fos = new FileOutputStream(f);
        PrintWriter printwrite = new PrintWriter(fos);

        String state = "Normal Display";
        String innerState = "Time";
        int m = 0, h = 0, D = 1, M = 1, Y = 2000;
        myInput.toLowerCase();
        for (int i = 0; i < myInput.length(); i++) {
            char currentChar = myInput.charAt(i);
            switch (state) {
                case "Normal Display":
                    if (currentChar == 'c') {
                        state = "Update";
                        innerState = "min";
                    }
                    if (currentChar == 'b') {
                        state = "Alarm";
                        innerState = "Alarm";
                    }
                    if (currentChar == 'a') {
                        if (innerState == "Time") {
                            innerState = "Date";
                        } else {
                            innerState = "Time";
                        }
                    }
```

```
                    if (currentChar == 'd') {
                        output.add("No action in this state with input d");
                    }
                    break;
                case "Alarm":
                    if (currentChar == 'a') {
                        if (innerState == "Alarm") {
                            innerState = "Chime";
                        }
                    }
                    if (currentChar == 'b') {
                        output.add("No action in this state with input b");


                    }
                    if (currentChar == 'c') {
                        output.add("No action in this state with input c");


                    }
                    if (currentChar == 'd') {
                        state = "Normal Display";
                        innerState = "Time";
                    }
                    break;
                case "Update":
                    if (currentChar == 'a') {
                        switch (innerState) {
                            case "min":
                                innerState = "hour";
                                break;
                            case "hour":
                                innerState = "day";
                                break;
                            case "day":
                                innerState = "month";
                                break;
                            case "month":
                                innerState = "year";
                                break;
                            case "year":
                                state = "Normal Display";
                                innerState = "Time";
                                break;
                        }
                    }
                    if (currentChar == 'b') {
                        switch (innerState) {
                            case "min":
                                m++;
                                if (m == 60) {
                                    m = 0;
                                    h++;
                                    if (h == 24) {
                                        h = 0;
                                        D++;
                                        if (D == 31) {
```

```
                                    D = 1;
                                    M++;
                                    if (M == 13) {
                                        M = 1;
                                        Y++;
                                    }
                                }

                            }
                        }
                    case "hour":
                        h++;
                        if (h == 24) {
                            h = 0;
                            D++;
                            if (D == 31) {
                                D = 1;
                                M++;
                                if (M == 13) {
                                    M = 1;
                                    Y++;
                                }
                            }

                        }
                    case "day":
                        D++;
                        if (D == 31) {
                            D = 1;
                            M++;
                            if (M == 13) {
                                M = 1;
                                Y++;
                            }
                        }
                    case "month":
                        M++;
                        if (M == 13) {
                            M = 1;
                            Y++;
                        }
                    case "year":
                        Y++;

                }
            }
            if (currentChar == 'c') {
                output.add("No action in this state with input c");

            }
            if (currentChar == 'd') {
                state = "Normal Display";
                innerState = "Time";
            }
            break;

        }
```

```
            output.add("Current State is : " + state);
            output.add("Current innerState is : " + innerState);
            output.add("DATE: " + Y + " - " + M + " - " + D);
            output.add("TIME: " + h + " : " + m);


        }
        System.out.println(output);
        printwrite.write(String.valueOf(output));
        printwrite.flush();
        fos.close();
        printwrite.close();



    }
}
```

- NB : I renamed my problems class to State for testing purposes

Test cases:

```
/*Tests for question 3 */


@Test
public void Test1() throws IOException {
    State test = new State();
    test.setMyInput("abbcd");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test1.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test2() throws IOException {
    State test = new State();
    test.setMyInput("Aa");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test2.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}
```

```java
@Test
public void Test3() throws IOException {
    State test = new State();
    test.setMyInput("aaa");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test3.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test4() throws IOException {
    State test = new State();
    test.setMyInput("cdb");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test4.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);

}

@Test
public void Test5() throws IOException {
    State test = new State();
    test.setMyInput("");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test5.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test6() throws IOException {
    State test = new State();
    test.setMyInput("caaaa");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test6.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test7() throws IOException {
    State test = new State();
    test.setMyInput("ba");
```

```
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test7.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test8() throws IOException {
    State test = new State();
    test.setMyInput("cb");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test8.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test9() throws IOException {
    State test = new State();
    test.setMyInput("cab");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test9.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test10() throws IOException {
    State test = new State();
    test.setMyInput("caab");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test10.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test11() throws IOException {
    State test = new State();
    test.setMyInput("caaab");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test11.txt");
    String contentOf_expectedOutput_file =
```

```
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}

@Test
public void Test12() throws IOException {
    State test = new State();
    test.setMyInput("caaaab");
    test.State();
    Path realOutput_file = Path.of("Output.txt");
    String contentOf_realOutput_file = Files.readString(realOutput_file);
    Path expectedOutput_file = Path.of("Test12.txt");
    String contentOf_expectedOutput_file =
Files.readString(expectedOutput_file);
    assertEquals(contentOf_expectedOutput_file, contentOf_realOutput_file);
}
```

Expected output textfiles:

Test Case 1 expected output file:

[Current State is : Normal Display, Current innerState is : Date, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Alarm, Current innerState is : Alarm, DATE: 2000 - 1 - 1, TIME: 0 : 0, No action in this state with input b, Current State is : Alarm, Current innerState is : Alarm, DATE: 2000 - 1 - 1, TIME: 0 : 0, No action in this state with input c, Current State is : Alarm, Current innerState is : Alarm, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Normal Display, Current innerState is : Time, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 2 expected output file:

[Current State is : Normal Display, Current innerState is : Time, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Normal Display, Current innerState is : Date, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 3 expected output file:

[Current State is : Normal Display, Current innerState is : Date, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Normal Display, Current innerState is : Time, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Normal Display, Current innerState is : Date, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 4 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Normal Display, Current innerState is : Time, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Alarm, Current innerState is : Alarm, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 5 expected output file:

[Your Input is empty]

Test Case 6 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : day, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : month, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : year, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 7 expected output file:

[Current State is : Alarm, Current innerState is : Alarm, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Alarm, Current innerState is : Chime, DATE: 2000 - 1 - 1, TIME: 0 : 0]

Test Case 8 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : min, DATE: 2001 - 2 - 2, TIME: 1 : 1]

Test Case 9 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2001 - 2 - 2, TIME: 1 : 0]

Test Case 10 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : day, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : day, DATE: 2001 - 2 - 2, TIME: 0 : 0]

Test Case 11 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : day, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : month, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : month, DATE: 2001 - 2 - 1, TIME: 0 : 0]

Test Case 12 expected output file:

[Current State is : Update, Current innerState is : min, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : hour, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : day, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : month, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : year, DATE: 2000 - 1 - 1, TIME: 0 : 0, Current State is : Update, Current innerState is : year, DATE: 2001 - 1 - 1, TIME: 0 : 0]

Test Cases Result: