Homework Assignment 3

Matthew O'Brien PSU ID: 965010701 email:matthewo@pdx.edu

Submission Date: 5/28/2015

Collaboration Acknowledgment: Andrew McCann

Solutions Chapter 6:

Question 11.

Consider the following code fragment:

```
m = 1;
q = 2;

while(m<n)
{
        m = m + 1;
        q = q*2;
}
```

Prove that this code fragment correctly computes $q = 2^2$ if n is an element of { 1, 2, 3, …..}

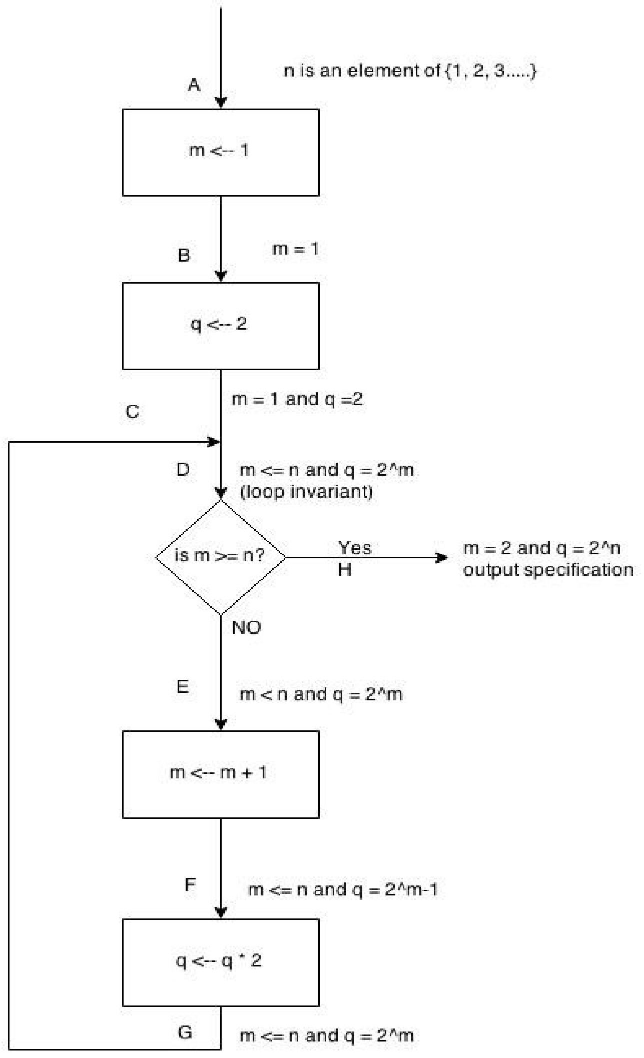
The following figure fully describes the code. Each element is marked A through G

A: n is an element of {1, 2, 3, ….}. This is the initial condition. Stating the same as the question.
B: Another starting condition, showing the beginning value of m = 1.
C: Starting condition, showing the beginning value of q = 2.
D: This is the loop variation for m <= n and $q = 2^m$ holding for the first round through the loop, because its the same as 2 to the first power = 1. Assuming that this holds for D at some $m = m_0$. If m_0 >= n, the loop will stop, and the output condition for H is met. Otherwise, E is next up, $m_0 < n$.
F: value of m_0 is incremented by 1, and m_0 <= n, and $q = 2^{m0 - 1}$ from the flowchart. At G, q is multiplied by 2. G follows from the fact that $2 \times 2^{2m0 - 1} = 2^{m0}$.

Note, the assertion of G is the same as the assertion of D. So the loop invariant will hold for all values of m such that, 1 <= m <= n.

The loop will terminate since m will continue to increment by 1 because of the m = m + 1, but m must be <= n. At which point, m will hit the value of n, and the loop is exited. And the value of q is still satisfied by the output specification the output of $q = 2^n$.

Figure on next page.



Question 14.

Design for the finite state machine.

States

S0 – Start state - Wait for first character of first word
S1 – Building the first word
S2 – Waiting for first character for next word
S3 – Building next word
S4 – Final State – End of file condition
S5 – Final State – Word is longer then the MAXPOS characters encountered

Inputs

NEWLINE – New line character
BLANK – Black character
EOF – End of file condition when read attempted
other – All other characters

Initial Conditions

Variable line_length, integer, number of characters in current output line. Starts at 0.
Variable word, character string, the word being built, initially empty.
Variable c, current input character, no initial value
Constant MAXPOS, maximum characters in the output line. Excludes the NEWLINE.

Transitions and Associated Actions(Note: the U is union, I couldn't get the actual symbol)

T1 – Wait for first word, and BLANK, or NEWLINE as input
T2 – Waiting for first word and EOF is reached.
T3 – Waiting for the first word and other character is input. Word ← c.
T4 – Building first word and BLANK or NEWLINE is input
    write(word)   line_length ← length(word)   word ← null
T5 – Building first word and EOF is reached.
T6 – Building first word and other character is reached.
    Word ← word U c (concatenate word word and c character c)
T7 – Building first word and its length will be > MAXPOS
T8 – Waiting for another word and BLANK or NEWLINE is input
T9 – Waiting for another word and EOF is reached
T10 – Waiting for another word and other character is input
    word ← word U c
T11 – Building another word and BLANK or NEWLINE is input.

Java Code:

```
import java.io.*;
import java.util.*

class NaurTextProcessing{
        private final int  MAXPOS = 10;
        private boolean debugOn = false;

        public void turnDebugOn(){
                debugOn = true;
        }

        public void turnDebugOff(){
                debugOff = false;
        }
```

```
private char getCharacter(RandomAccessFile inFile){
        try{
                char characterRead;
                if(inFile.getFilePointer() == inFile.length())
                        characterRead = '\0';
                else
                        charaterRead = (char) inFile.readByte();

                if(characterRead == ' ')
                        characterRead == ' ';
                return (characterRead);
        }
        catch(Exception e){
                System.out.println("ERROR!");
                System.out.println("\t" + e);
                return '\0';
        }
}

public void runApp(){
try{
        char currentChar;
        int lineLength = 0;
        char state = '0';
        StringBuffer word = new StringBuffer();
        char tmp;
        int I;
        File inputFile = new File("Naur.dat");

        if(!inputFile.exists()){
                System.out.println("File isn't there");
                return;
        }

        RandomAccessFile inFile = new RandomAccessFile(inputFile, "r");
        while(Character.digit(state, 10)<4){
                currentChar = getCharacter(inFile);
                if(debugOn){
                        System.out.println("State: " + state + "\t");
                        switch(currentChar){
                                case '\0';
                                System.out.println("End of file.");
                                break;
                                case' ':
                                        System,out.println("Blank Return")
                                        break;
                                default:
                                        System.out.println("Character read: " +currentChar + " ".);
                                }
```

```
    }
if switch(state){
    case: '0':
        if(debugOn)
            System.out.println("Entering State 0");
    switch(currentChar){
        case: ' ':
            break;
            state = '4';
        case: '\0':
            state = '4';
            break;
        default:
            word.append(currentChar);
            state = '1';
            break;
        case: '1'
            if(debugOn)
                System.out.println("enter state 1")
            switch(currentChar)
                case ' ':
                    System.out.print(word.toString()):
                    lineLength = word.length();
                    word = new StringBuffer();
                    state = '2';
                    break;
        case '\0':
            System.out.println(word.toString());
            state = '4';
            break;
            default:
                if(word.length() <= MAXPOS)
                    word.append(currentChar);
                else
                    state = '5'
                    break;
            }
    switch(currentChar)
        case '2':
            if(debugOn)
                System.out.println("Entering State 2")
    switch(currentChar){
        case ' ':
            break;
        case '\0':
            state = '4';
            break;
        default:
            word.append(currentChar)
```

```
                        state = '3';
                        break;
                }
                switch(currentChar)
                        case '3':
                                if(debugOn)
                                        System.out.println(“Entering 3”)
                                switch(currentChar){
                                        case ' ':
                                        case '\0':
                                        if((lineLength + word.length() <= MAXPOS)
                                                System.out.print(currentChar);
                                        System.out.print(word.toString());
                                        lineLength = lineLength + 1 + word.length();
                                        word = new StringBuffer();
                                }
                                else
                                        System.out.println();
                                        System.out.print(word.toString());
                                        lineLength = word.length();
                                        word = new Stringbuffer();
                }
                if(current != '\0')
                        state = '2';
                else
                        state = '4';
                break;
                default:
                        if((word.length())<= MAXPOS)
                                word.append(currentChar);
                        else
                                state = '5';
                break;
                switch(currentChar)
                        break;
                default:
                        break;
                }
        switch(state)}
        while(Character.digit(state, 10)<4)
                System.out.print();
        if(state == '5');
                System.out.println(“Word too long”)
        inFile.close();
}
catch(Exception e){
        System.out.println(“Error in the main”);
        System.out.println(“\t” + e);
}
```

There were quite a few faults I found with my implementation. I attempted to make something a little more exact and precise but in turn made it very confusing. This the question says to leave the code how it is, and not fix any of my problems, I did just that. But there were quite few issues with my solution. This mostly came from my attempt to use cases, breaks, and switches which were all new to me. The question also asked to compare with another student. I did not actually catch that part, so I didn't have anyone look at my code.

<u>Solutions Chapter 15:</u>

Question 15:

1.Test case: BLANK (T1)
Expected output: NEWLINE

2.Test case: Empty File (T2)
Expected output: NEWLINE

3.Test case: NEWLINE(T1)
Expected output: NEWLINE

4.Test case: c (T3, T5)
Expected output: NEWLINE

5.Test case: c BLANK (T4, T9)
Expected output: c NEWLINE

6.Test case: c c (T6)
Expected output: c c NEWLINE

7.Test case: c+, length > MAXPOS (T7)
Expected out: Error.

8.Test case: c+ BLANK BLANK (T8)
Expected out: c+ NEWLINE

9.Test case: c+ BLANK (T9)
Expected out: c+ NEWLINE

10.Test case:c+ BLANK c, length <= MAXPOS (T10, T12)
Expected out: c+ BLANK c NEWLINE

11.Test case: c+ BLANK c BLANK, length <= MAXPOS (T11)
Expected output: c+ BLANK c NEWLINE

12.Test case: c+ BLANK c BLANK, length > MAXPOS (T11)
Expected out: c+ NEWLINE c NEWLINE

13.Test case: c+ BLANK c, length > MAXPOS (T12)
Expected out: c+ NEWLINE c NEWLINE

14.Test case: c+ BLANK c c (T13)
Expected out: c+ BLANK c BLANK c c NEWLINE

15.Test case: c+ BLANK c+, length of second c+ > MAXPOS (T14)
Expected out: Error.

Question 16.

Since the solution to problem 5.14 is a very tree-like structure already, the branch-coverage test cases are the same for this problem as the previous problem. There is no need to do it all again.

Question 17.

The variable state being defined from the beginning of the program with the while statement. Therefore, every test case will use that path. And the value of that state is changed with every transition except for T1, T6, T 8, and T13, after which the while takes over again, and uses that value. The variable of currentChar is from the function getCharacter, and was used for all the transitions, the switch(currentChar). This allows for the same test cases from question 15 again.

The "cases" correspond to the numbered cases from question 15.

Test cases:

T4:

Used in T11	Cases 10, 11
Used in T12	Cases 12, 13

Defined in T11:

Used in T11	Test Case: c+ BLANK c BLANK c BLANK
	Out: c+ BLANK c Blank c NEWLINE
Used in T12	Test: c+ BLANK c BLANK c
	Out: c+ BLANK c Blank c NEWLINE

Defined in T12:
Program will terminate here. For a variable word, it will be defined before the while loop, and the following transitions of T3, T4, T6, T10, T11, and T13 used the same transitions. Same with T14.
Defined in T3 and T4: Both covered by previous cases.

Defined in T6:

Used in T3	Error
Used in T4	Test: c c Blank
	Out: c c NEWLINE
Used in T6	Test: c c c
	Out: c c c NEWLINE
Used in T10	Test: c c BLANK c
	Out: c c BLANK c NEWLINE
Used in T11	Test: c c BLANK c BLANK
	Out: c c BLANK c NEWLINE
Used in T13	test: c c BLANK c c
	Out: c c BLANK c c NEWLINE
Used in T14	Test: c c BLANK c c+, length>MAXPOS
	out: Error

Defined in T10:

Used in T3, T4, T6:	Impossible
Used in T10	Test: c+ BLANK c BLANK c
	Out: c+ BLANK c Blank c NEWLINE
Used in T11	Test: c+ blank c blank
	out: c+ blank c newline
Used in T13	test: c+ blank c c
	out: c+ blank c c newline
Used in T14	Test: c+ blank c+, length > maxpos
	Error

Defined in T11:

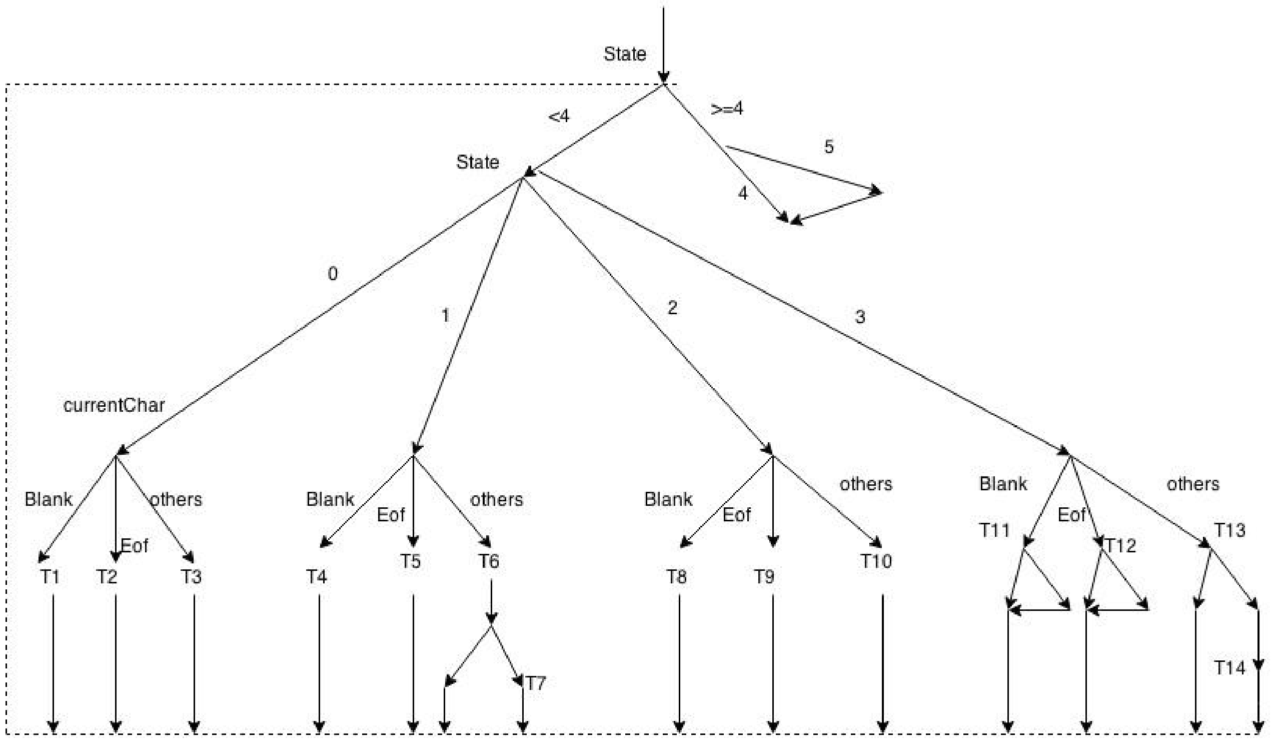
Used in T3, T4, T6:	Impossible
Used in T10	Test: c+ BLANK c BLANK c
	Out: c+ BLANK c Blank c NEWLINE
Used in T11	Test: c+ BLANK c BLANK c BLANK
	Out: c+ BLANK c Blank c NEWLINE
Used in T13	Test: c+ BLANK c BLANK c c
	Out: c+ BLANK c BLANK c c NEWLINE
Used in T14	Test: c+ BLANK c BLANK c+, length>MAXPOS
	out: Error

Defined in T13:

Used in T3, T4, T6:	Impossible
Used in T10	Test: c+ BLANK c c BLANK c
	Out: c+ BLANK c c BLANK c NEWLINE
Used in T11	Test: c+ BLANK c c BLANK
	Out: c+ BLANK c c NEWLINE
Used in T13	Test: c+ BLANK c c c
	Out: c+ BLANK c c c NEWLINE
Used in T14	Test: c+ BLANK c+, length > MAXPOS
	Out: Error.

Question 18:

Since the solution is in a tree like structure, all test cases from question 15 are the same for this answer. The following flowchart will better explain this though.



Question 19:

The previous flowchart can be used can used to describe this problem as well. The only changes made are L = { start, while statement, switch statements, all case statements, default statements, all if statements, all else statements, all break statements, end} since this is for a linear sequence of coding.

Question 20:

Refer the the flowchart from question 18. From the flowchart it shows that e = 4 + 31, n = 3 + 15, c = 2. The cyclomatic number M = 35 – 18 + 2 x 2 = 21.

The number of edges is e, the nodes are n, and connected are c. Using the equation M = e – n + 2c.