75

GRADUATE OUTREACH PROCTOR

Test Instructions and Signature Page (Please return with completed exam)

Semester:	Spring 2017
Course Name:	Software Quality Assurance
Course Number:	COMP 6716
Exam Number:	2
Professor:	Cross
Phone:	888-844-5300

Proctor: **Andrew Ryan**

Student: Robin Ward

NO COPIES OF COMPLETED EXAM ARE TO BE GIVEN TO STUDENT

STUDENT IS RESPONSIBLE FOR COST OF MAILING EXAM BACK

THE PROCTOR, NOT THE STUDENT, MUST MAIL THE EXAM

Time Allotted for Exam: 1 hour 15 mins

Instructions for Administering Test: Student is allowed a writing instrument. Closed book, closed notes. Please instruct student to place their answers directly on the exam.

Exam return due date: Wednesday, March 29, 2017 via email and mail hardcopy; STUDENT is responsible for contacting Professor if deadline cannot be met.

THE SIGNATURE OF THE PROCTOR LISTED ABOVE IS REQUIRED FOR ALL EXAMS

Please explain below if someone other than the above proctor administers exam

---

I certify that the instructions were followed and I proctored the exam as stated.

Proctor's Signature Andrew Ryan Date Exam was Administered 3/23/17

EMAIL & MAIL exam & this SIGNED page to:

April Hurley
Engineering & Business Online
202 Ramsay Hall
261 W. Magnolia Avenue
Auburn University, AL 36849-5336
Email: ash0008@ auburn.edu
(888) 844-5300

75

COMP 5710, 6710, 6716
Software Quality Assurance – Exam 2

Name: Robin Ward Date: March 22, 2017

100 pts

If you need extra space for your answers, please use the back of the page and indicate that you did so.

(5) 1. Differentiate between *white box testing* and *black box testing*. Also, include other names for these. Black Box testing AKA functional testing is testing of the actual program and verifying the output from the input. No code is examined during this test. White Box AKA Structural testing, is examining the code and making sure that it is written properly.

(4) 2. Describe what is meant by the term *independent path* (or basis path).

-2 That is a path that has a single point of entry and exit. This would be one of the paths in a program.

(2) 3. How is the *cyclomatic complexity* of a program related to the number of independent paths in the basis set?

The cyclomatic complexity number is equal to the amount of (independent) paths in a program.

(6) 4. Describe three ways to compute the *cyclomatic complexity* of a program.

1. $|E| - |V| + 2$
2. Regions + 1
3. Conditions + 1

(3) 5. To what extent is a *loop* in a program tested by a set of basis paths?

-1 it is tested via a specified amount of iterations

-3

(20) 6. If we define P* to be the total number of syntactic paths defined by a unit of code, calculate P* for the following source code for each of the assumptions given below in a, b, and c. Assume there is only one path for every S-statement. Be sure to show the expression you used to arrive at your answer. Drawing a control flow graph may be helpful, especially with respect to the multiple conditions in the if and else if.

```
1   pathArithmetic()
2   {
3     S1;
4     if (C1 && C2 && C3)
5       S2;
6     else if (C4 || C5)
7       S3;
8     else
9     {
10      for (int i = 0; C6; i++)
11        if (C7)
12          S4;
13        else
14          S5;
15    }
16    S6;
17    while (C8)
18      S7;
19    S8;
20  }
```

Sequence - Multiply
Decision - Add
Iteration - exponential

0, 1, 2

$1^0 + 1^1 + 1$

(15 pts) Assumptions for calculating P*:

a. (5 pts) The *for* loop has exactly 4 iterations; the *while* loop has 0, 1, or 2 iterations.

-3 P* = $1 \times (2+3=5 + 4^2) + 0^0 + 0^1 + 0^2 \times 1 = 22$

2^1 $1 \times 21 + 1 \times 1 =$

b. (5 pts) The *for* loop has exactly 3 iterations; the *while* loop has 0, 1, 2, or 3 iterations.

-3 P* = $1 \times (2+3=5 + 3^2) = 14$

$9+9=14$

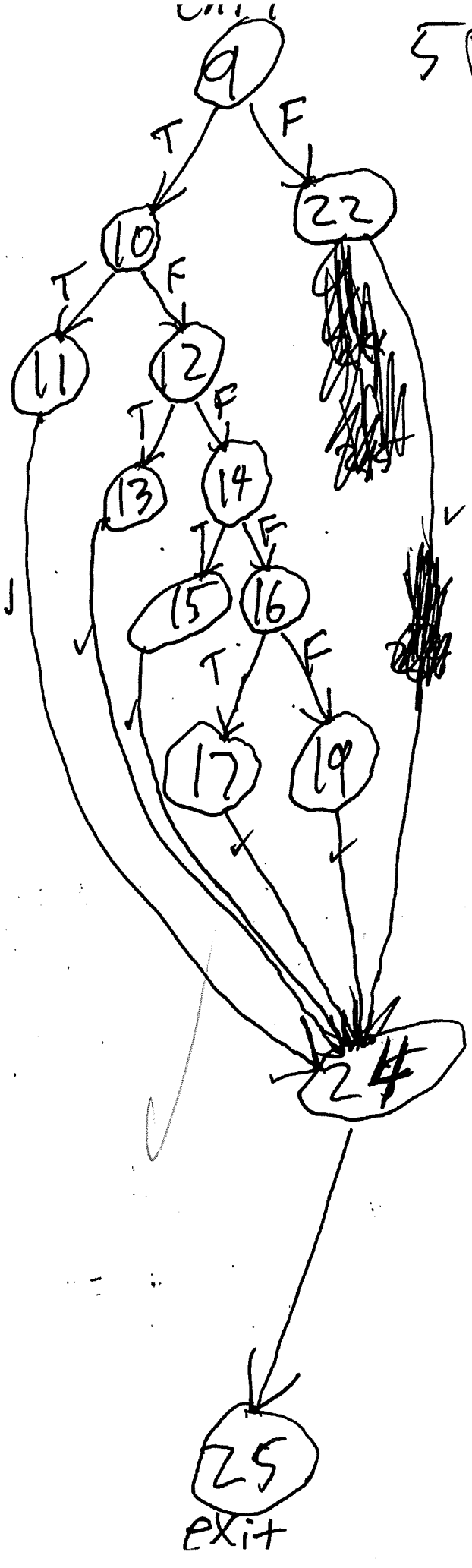
c. (5 pts) The *for* loop has exactly 2 iterations; the *while* loop has exactly 10 iterations.

-2 P* = $1 \times (2+3=5) + 4 = 1 \times 9 \times 1 \times 1 \times 1 = 9$

(5 pts) What is the cyclomatic complexity this program? Show your calculations.

Conditions $+1 = 8 + 1 = 9$

-8



5 Regions
+
1
=
6 VG

(30) 7. Basis paths, path predicates – Consider the program below that determines the total cost of dining out including tax and tip.

```
1   public static void main(String[] args) {
2     int guests;
3     double foodTotal, taxRate, totalWithTax, finalTotal, tipRate;
4     Scanner scan = new Scanner(System.in);
5     System.out.println("Enter number of guests, food total, and tax rate");
6     guests = scan.nextInt(); foodTotal = scan.nextDouble();
7     taxRate = scan.nextDouble();
8     totalWithTax = foodTotal * (1 + taxRate);
9     if (guests < 6) {
10      if (totalWithTax < 5.0)
11        tipRate = 0.35;
12      else if (totalWithTax < 20.0)
13        tipRate = 0.30;
14      else if (totalWithTax < 50.0)
15        tipRate = 0.25;
16      else if (totalWithTax < 100.0)
17        tipRate = 0.20;
18      else
19        tipRate = 0.18;
20    }
21    else
22      tipRate = 0.15;
23
24    finalTotal = totalWithTax * (1 + tipRate);
25    System.out.println("Final Total: " + finalTotal);
26  }
```

a. (12 pts) Draw the control flow graph using line numbers (and when necessary, conditions of T or F) beginning at line 9.

- graph on back of previous page

b. (18 pts) Based on the flow graph (or program), derive a set of basis paths and path predicates. For the path predicate use T, F, and X for *true*, *false*, and *don't care* respectively.

#	Path	Path Predicate line #s: 9	10	12	14	16
1	1-2-3-4-5-6-7-8-9-22-24-25	F	X	X	X	X
2	1-2-3-4-5-6-7-8-9-10-11-24-25	T	T	X	X	X
3	1-2-3-4-5-6-7-8-9-10-12-13-24-25	T	F	T	X	X
4	1-2-3-4-5-6-7-8-9-10-12-14-15-24-25	T	F	F	T	X
5	1-2-3-4-5-6-7-8-9-10-12-14-16-17-24-25	T	F	F	F	T
6	1-2-3-4-5-6-7-8-9-10-12-14-16-19-24-25	T	F	F	F	F

(30) *8.* Definition-Use paths – For the following program, find all of the definitions, uses, and D-U paths for the six variables indicated in the table below.

```
1  public static void main(String[] args) {
2     int guests;
3     double foodTotal, taxRate, totalWithTax, finalTotal, tipRate;
4     Scanner scan = new Scanner(System.in);
5     System.out.println("Enter number of guests, food total, and tax rate");
6     guests = scan.nextInt(); foodTotal = scan.nextDouble();
7     taxRate = scan.nextDouble();
8     totalWithTax = foodTotal * (1 + taxRate);
9     if (guests < 6) {
10       if (totalWithTax < 5.0)
11          tipRate = 0.35;
12       else if (totalWithTax < 20.0)
13          tipRate = 0.30;
14       else if (totalWithTax < 50.0)
15          tipRate = 0.25;
16       else if (totalWithTax < 100.0)
17          tipRate = 0.20;
18       else
19          tipRate = 0.18;
20    }
21    else
22       tipRate = 0.15;
23
24    finalTotal = totalWithTax * (1 + tipRate);
25    System.out.println("Final Total: " + finalTotal);
26 }
```

Use line numbers from the program to complete the table below.

Variable	Definitions	Uses	Definition-Use Paths (separated by ';')
guests	2, 6	9	~~2-3-4-5-6;~~ 2-3-4-5-6-7-8-9
foodTotal	3, 6	8	~~3-4-5-6; 3-4-5-6-7-8~~
taxRate	3, 7	8	~~3-4-5-6-7; 3-4-5-6-7-8~~
tipRate	3, 11, 13, 15, 17, 19, 22 24	9-10-11-24; 9-10-12-13; 9-10-12-14-15; 9-10-12-14-16-17; 9-10-12-14-16-19; 9-22-24	
totalWithTax	3, 8,	10, 12, 14, 16, 24	9-10; 9-10-12; 9-10-12-14; 9-10-12-14-16; 9-22-24
finalTotal	3, 24	25	24-25

all must begin at 8 (-2)

5 missing (-5)

(100) *Total points on Exam 2*

-14