Cairo University

Faculty of Computers and Information

Computer Science Department

CS 496: Software Testing, MIDTERM EXAMINATION

F 2015

STUDENT NAME _____

STUDENT ID # _____

- Put your name and ID number on the examination booklet.
- Answer all questions in the examination booklet.
- The exam is CLOSED BOOK, no books, no notes, no calculators, no electronic devices allowed.
- Total marks available: 75 (equivalent to 15% of your course's total grade).
- **This booklet consists of 8 pages (including this cover page).**
- All inquiries and requests must be address to the supervisors only.
- **The following is strictly prohibited:**
    - speaking to other students or communicating with them under any circumstances whatsoever;
    - bringing into the exam any textbook, notebook or document not authorized by the examiner;
    - making use of calculators, cameras, cell-phones, computers, head-sets, pagers, PDA's, or any device not authorized by the examiner;
    - leaving answer papers exposed to view;
    - attempting to read another student's examination papers.
- Students must stop writing when the signal is given. Answer books must be handed to the supervisor-in-charge promptly. Failure to comply with these regulations will be cause for rejection of an answer paper.
- Duration: 60 minutes

Do NOT open the booklet until instructed to do so.

Question 1 [16 marks]

We begin with some basics regarding software testing:

a) What is a test case? What are its main components? **[4]**

A test case is a test-related item which contains the following information: a set of test inputs, execution conditions, and expected outputs.

b) Define the terms test suite, and test oracle. **[4]**

Test suite: A test suite is a group of related test cases.

Test oracle: A test oracle is some means that allows testers to determine whether a test has passed or failed, by checking whether the observed and expected behaviours match or not.

c) During the lectures, we talked about a set of testing principles. Consider the following scenario, and explain what testing principles has been followed, and what testing principles have been violated. For each case, you need to **mention the testing principle,** and **explain how it has been followed/violated. [8]**

David works as a software developer in some software company. He has just finished developing code to automate the process of withdrawing money from an Automated Teller Machine (ATM). Such code needs to be tested. Hence, David created JUnit tests that covered two scenarios:

(i) Allowing a customer to withdraw 40 dollars from his account that has a 100 dollars balance.

```
public void testWithdraw_scenario1()
{
     Customer John = new Customer(100);
     //where 100 is the initial balance

     int newBalance = John.withdraw(40);
     assertEquals(newBalance, 60);
}
```

(ii) Allowing a customer to withdraw 100 dollars from his account that has a 100 dollars balance.

```
public void testWithdraw_scenario2()
{
     Customer John = new Customer(100);
     //where 100 is the initial balance
```

```
        John.withdraw(100);
    }
```

Followed principles:

- Used JUnit which is a reusable approach to write tests.

Violated principles:

- He wrote the source code, and tested for it. This violates the principle that one should not test his own code.
- He wrote test cases for the valid and expected inputs (i.e., no handling for negative amounts, or for withdrawing extra money for instance). This violates the principle that test cases should consider both the valid and invalid inputs/cases.
- The absence of any definition/assertion of the expected result within the second test case.

Question 2 [34 marks]

We have looked at a number of black-box techniques for testing.

a) Explain what black-box testing and white-box testing mean, and the difference between them. **[4]**

Within black box testing, the tester does not have access to the source code, nor to how the software operates. He only knows what the software is supposed to do.

Within white-box testing, the tester has access to the program's source code, hence using the code's structure while designing his test cases.

b) `NextDate` is a function that takes three arguments as input: month, day, year. It has the following specifications:

- It returns the date of the day following the input date. The allowed years are from 1812 – 2020.
- If it is not the last day of the month, only the day value will be incremented.
- At the end of a month, the next day is 1 and the month is incremented.
- At the end of the year, both the day and the month are reset to 1, and the year gets incremented.
- Leap year (سنة كبيسة) definition: A 29th day is added to February in all years that are evenly divisible by 4, except for centennial years (i.e., years that end with –00) which are not evenly divisible by 400. Hence, 1600, 2000 and 2400 are leap years, but 1700, 1800, 1900, 2100, 2200 and 2300 are not.

Identify the equivalence partitions for this function. **Note that you are not required to create concrete test cases.** You only need to generate the equivalence partitions. **[18]**

Month:

- **E1:** month has 30 days, **E2**: month has 31 days, **E3**: February, **E4** >=13,
- **E5**: <=0, **E6**: empty, **E7**: any non-integer

Day

- **E1**: 1<=day <= 28, **E2**: 1<=day <=29, **E3**: 1<=day <= 30, **E4**: 1<=day <= 31, **E5:** day >= 32, **E6**: day<=0, **E7**: empty, **E8**: any non-integer

OR **E1**: 1<=day<=28, **E2**: day = 29, **E3**: day = 30, **E4**: day =31, **E5**: day >=32, **E6**: day<=0, **E7**: empty, **E8**: any non-integer

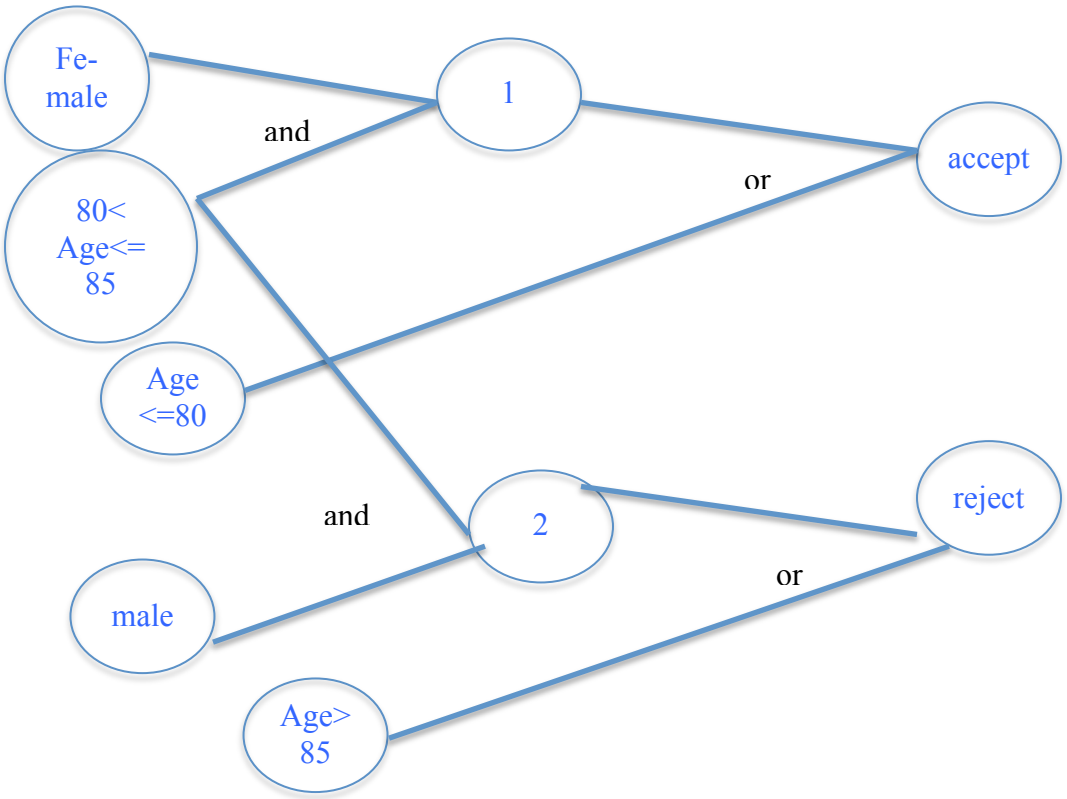
Year

- **E1**: year=1900;
- **E2**: (1812<=year <= 2020) AND (year != 1900) AND (year mod 4 = 0)
- **E3**: (1812<=year <= 2020) AND (year != 1900) AND (year mod 4 != 0)
- **E4**: year < 1812, **E5**: year > 2020, **E6**: Any non-integer, **E7**: empty

c) Consider an insurance system that allows users to apply for insurance on their lives, and rejects over-age applicants. The system has the following specifications:

- Reject male applicants if over the age of 80 years.
- Reject female applicants if over the age of 85 years.
- Accept applicants otherwise.

Build a cause-effect graph for that system. **[12]**



Question 3 [20 marks]

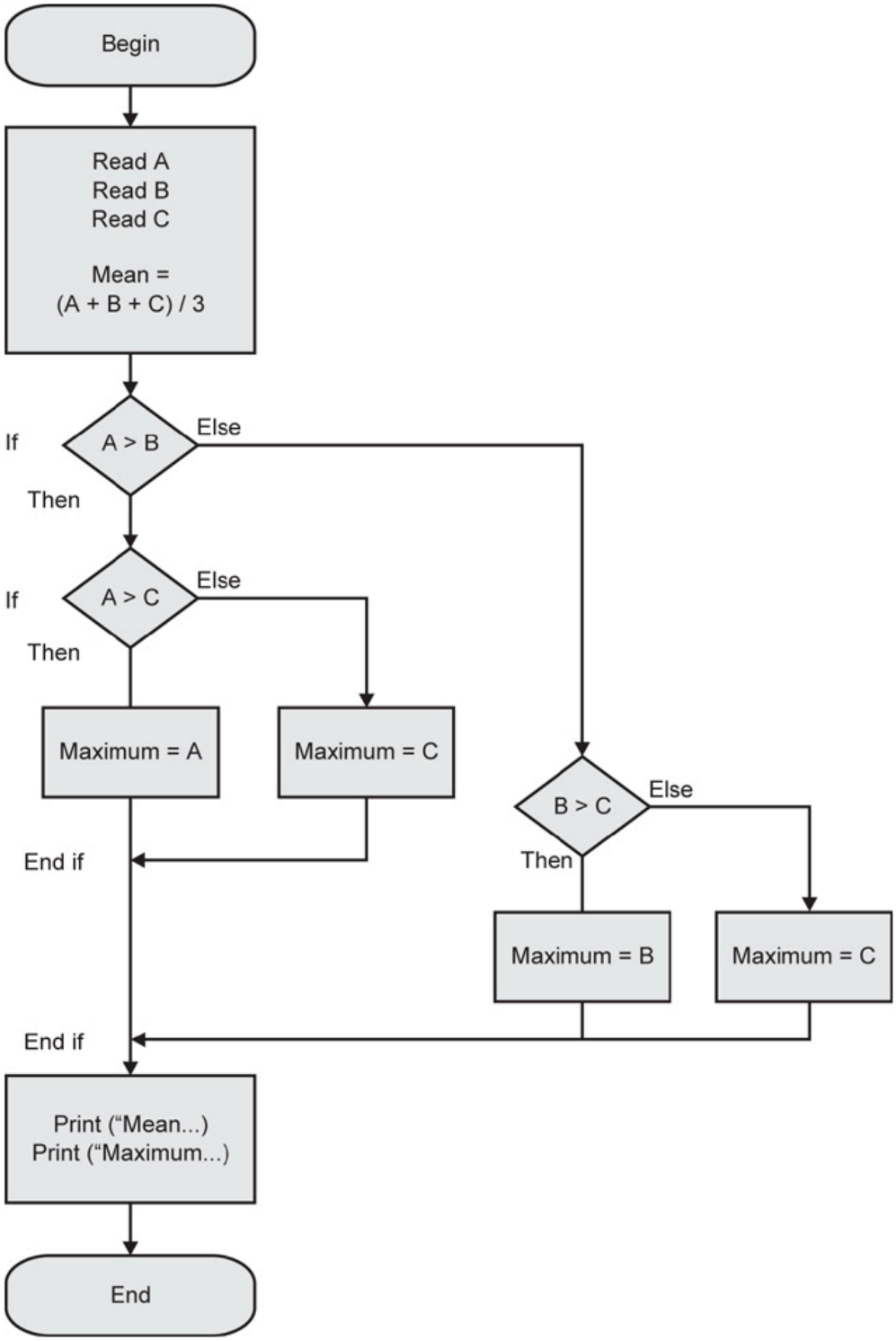
We have looked at a number of white-box testing techniques.
a) Create a Control Flow Graph (CFG) for the following code **[8]**:

```
Program MaxAndMean
   Read A
   Read B
   Read C
   Mean = (A + B + C)/3

   If A > B
   Then
        If A > C
        Then
             Maximum = A
        Else
             Maximum = B
        Endif
   Else
        If B > C
        Then
             Maximum = B
        Else
             Maximum = C
        Endif
   Endif

   Print("Mean of A, B, and C is ", Mean)
   Print("Maximum of A, B, and C is ", Maximum)

End
```


Begin
Read A
Read B
Read C
Mean =
(A + B + C) / 3
If
A > B
Else
Then
If
A > C
Else
Then
Maximum = A
Maximum = C
End if
B > C
Else
Then
Maximum = B
Maximum = C
End if
Print ("Mean...)
Print ("Maximum...)
End

b) Consider the following code. Given a test suite T that has two tests t1, and t2 as follows: t1: <x=1, y=0>, t2: <x= 2, y=1>.

Calculate the branch (i.e., decision) coverage of that the test suite T. You need to show all the steps for your calculation **[8]**:

```
Begin

   int x,y,z;
   input(x,y);

   if(y > 0)
   {
        if( x=0)
             z = foo1(x,y);
        else
             z = foo2(x,y);

        y = y-1;

   }
   else
   {
        z = foo3(x,y);
   }

 End
```

T1:<x=1, y=0>:
   Covers y>0 → false branch
T2: <x=2, y=1>:
   Covers y>0 → true branch
   Covers x=0 → false branch

Branch coverage = ¾=0.75

Tracing for T1
Tracing for T2
Equation for branch coverage
Substitution to get the correct coverage

c) Consider the following code. Derive a number of tests to achieve 100 percent statement coverage **[4]**.

```
Begin

   int Interest = 0.035;
   int Balance;
   input(Balance);

   If Balance > 1000
   Then
        Interest = Interest + 0.005;
        If Balance < 10000
        Then
             Interest = Interest + 0.005;
        Else
             Interest = Interest + 0.010;
        Endif
   Endif

   Balance = Balance * (1 + Interest);

End
```

T1: balance = 5000
T2: balance = 20000

Question 4 [5 marks]

Considering the agile model explained within your presentations, define the terms scrum, stand-ups, and sprint. **[3]**

- Scrum: One of the agile software development methodologies.
- Daily Scrum/Stand-ups : Each day of a sprint, the team holds a daily scrum (or a stand-up) to monitor the progress, identify risks, and share knowledge.
- Sprint: It is the iteration of the scrum methodology including the design, building, integration, documentation, and testing. The recommended duration of a sprint is 4 weeks.

Briefly explain the software development life cycle within the agile model. **[2]**

See your colleagues' presentation on the agile and TDD for the explanation.

End of Exam