Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto, Raffaela Mirandola

20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Software Engineering II

July 9th 2015

Last Name
First Name
Id number (Matricola)

Note

1. The exam is not valid if you don't fill in the above data.
2. Write your answers on these pages. Extra sheets will be ignored. You may use a pencil.
3. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
4. You cannot keep a copy of the exam when you leave the room.

Question 1 Alloy (7 points)

A grocery store offers to its customers some barcode readers that allow them to read the barcodes of the products they want to buy and add the corresponding cost to their bill.
The barcode readers are located in a suitable dispenser and can be taken by customers only if their batteries are sufficiently charged.

1. Define an Alloy model for the description above
2. Model the operation getBarCodeReader that extracts the reader from the dispenser and associates it to a customer

Solution
Below you find a straightforward solution. Others are possible.

```
sig Barcode {}
sig BatteryLevel {}
sig Low extends BatteryLevel {}
sig Price {}

sig BarcodeReader{
 bl: one BatteryLevel,
 bill: set Price,
 currentProduct: Product
}

sig Dispenser {
 brSet: set BarcodeReader
}

sig Customer {
 br: lone BarcodeReader
}

sig Product {
 p: Price,
 b: Barcode
}

fact barcodeUnique { no disjoint p1, p2: Product | p1.b = p2.b}
fact noTwoCustomerSharingBarcodeReader {no disjoint c1, c2: Customer | c1.br = c2.br}

pred getBarCodeReader[d: Dispenser,  d': Dispenser, c: Customer] {
 some bbr: BarcodeReader | bbr in d.brSet and not (bbr.bl in Low) and let bbbr = bbr {

 d'.brSet = d.brSet - bbbr
 c.br = bbbr }
}

pred show[]{}

run show
run getBarCodeReader
```

Questions 2 Planning (5 points) and design (5 point)

We want to develop a system for the management of an association of companies related to a specific industrial sector. The system should be accessible via browser only for registered users.

-The association administrator should be able to manage the companies' registration fees (fee computation and invoice), see below for details. Moreover, he/she should be able to add to the system the information about the events promoted by the association (when the event is scheduled, where it will take place and the participation cost).
-The registered companies should be able to see their profile, update their information, check their payments, and register to the events promoted by the association.

Every year each company provides its sales volume and based on that, the association administrator computes the annual registration fee.

Every two months, the registered companies receive via mail the amount to be paid and the invoice. This amount includes the subscription fee for the two months and the cost of participation to events organized by the association (if any). The invoice details are available through the system.

Whenever the association receives the payment of an invoice, the system administrator marks it as paid.

Given the above description, do the following:

A) **Calculate the function points** for the system. Refer to the following table to associate weights to the function types:

Function types	**Weights**		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. Internal Files	7	10	15
N External Files	5	7	10

B) **Define the system architecture**. List of components belonging to the architecture and describe the function of each component and the interaction between components. Moreover, list the technologies you are going to use and provide a justification for your choice.

You can add any hypothesis or functionality you think is important to consider in this case, provided that you do so explicitly.

Question 4: Testing (5 points)

Consider the following fragment of a program using natural numbers (nat)

```
1. nat y, a, b;
2. read (a, b, y);
3. while (a < y) {
4.         if (a < b)
5.                   a = b;
6.         else if (a > b)
7.                    b = a;
8.              else a++; }
9. ...
```

You are to:

- Define the flow graph for the fragment above augmented with the annotations concerning the definition and usage of the variables
- Execute the program symbolically showing different possible execution paths involving the execution of statement 7.
- For each of these paths show the path condition and possible test cases.

Solution

A) Flow graph is quite basic, definition and usage of variables is as follows:
1, 2: Def a, b, c
3: Use a, y
4: Use a, b
5: Use b, Def a
6: Use a, b
7: Use a, Def b
8: Use a, Def a

B) Symbolic execution of possible paths involving statement 7

We try to identify those paths that pass through 7 and stay in the while for the smallest possible number of iterations:
Path 1
<1, 2, 3, 4, 6, 7, 3, 9> **Not possible, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and not(A<B)
6: a = A, b = B, y = Y, path condition: A<Y and A>B
7: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
3: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and not(A>Y), which is not possible

Path II
<1, 2, 3, 4, 6, 7, 3, 4, 6, 7, 3, …> **Not possible, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and not(A<B)
6: a = A, b = B, y = Y, path condition: A<Y and A>B

7: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
3: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
4: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and not(A<A)
6: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and not(A<A) and A>A, which is not possible

Path III
<1, 2, 3, 4, 6, 7, 3, 4, 5, 3, …> **Not possible, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and not(A<B)
6: a = A, b = B, y = Y, path condition: A<Y and A>B
7: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
3: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
4: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and A<A, which is not possible

Path IV
<1, 2, 3, 4, 6, 7, 3, 4, 6, 8, 3, 9> **, path condition A<Y and A>B and A+1==Y, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and not(A<B)
6: a = A, b = B, y = Y, path condition: A<Y and A>B
7: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
3: a = A, b = A, y = Y, path condition: A<Y and not(A<B)
4: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and not(A<A)
6: a = A, b = A, y = Y, path condition: A<Y and not(A<B) and A==A
8. a = A+1, b = A, Y = Y, path condition: A<Y and not(A<B)
3. a = A+1, b = A, Y = Y, path condition: A<Y and not(A<B) and A+1==Y
9. …

Path V
<1, 2, 3, 4, 5, 3, 4, 6, 7, 3, …> **Not possible, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and A<B
5: a = B, b = B, y = Y, path condition: A<Y and A<B
3: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y
4: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y and not(B<B)
6: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y and not(B<B) and B>B, which is not possible

Path VI
<1, 2, 3, 4, 5, 3, 4, 6, 8, 3, 4, 6, 7, 3, 4, 6, 8, 3, 9>, **path condition A<Y and A<B and B+2==Y, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and A<B
5: a = B, b = B, y = Y, path condition: A<Y and A<B
3: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y
4: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y and not(B<B)
6: a = B, b = B, y = Y, path condition: A<Y and A<B and B < Y and B==B
8: a = B+1, b = B, y = Y, path condition: A<Y and A<B and B < Y and B==B
3: a = B+1, b = B, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y
4: a = B+1, b = B, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y and not(B+1<B)

6: a = B+1, b = B, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y and B+1>B
7: a = B+1, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y
3: a = B+1, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y
4: a = B+1, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y and not(B+1<B+1)
6: a = B+1, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y and B+1==B+1
8: a = B+2, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+1<Y
3: a = B+2, b = B+1, y = Y, path condition: A<Y and A<B and B < Y and B+2==Y

Path VII
<1, 2, 3, 4, 6, 8, 3, 4, 6, 7, 3, 4, 6, 8, 3, 9>, **path condition A<Y and A==B and A+2==Y, in fact:**
1, 2: a = A, b = B, y = Y, path condition: true
3: a = A, b = B, y = Y, path condition: A<Y
4: a = A, b = B, y = Y, path condition: A<Y and not(A<B)
6: a = A, b = B, y = Y, path condition: A<Y and A==B
8: a = A+1, b = B, y = Y, path condition: A<Y and A==B
3: a = A+1, b = B, y = Y, path condition: A<Y and A==B and A+1<Y
4: a = A+1, b = B, y = Y, path condition: A<Y and A==B and A+1<Y and not(A+1<B)
6: a = A+1, b = B, y = Y, path condition: A<Y and A==B and A+1<Y and A+1>B
7: a = A+1, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y and A+1>B
3: a = A+1, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y
4: a = A+1, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y and not(A+1<A+1)
6: a = A+1, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y and A+1==A+1
8: a = A+2, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y
3: a = A+2, b = A+1, y = Y, path condition: A<Y and A==B and A+1<Y and A+2==Y

Other paths could be analysed but we have identified the three ones that pass through 7 and stay in the while for the smallest possible number of iterations.
Note: the text is not asking you to be exhaustive. Even a smaller but meaningful number of path analyses would be acceptable.

C) Test cases descend from the three path conditions, for instance:
a = 2, b = 1, y = 3
a = 1, b = 2, y = 4
a = 2, b = 2, y = 4