Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto and Matteo Rossi

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

Software Engineering 2 – Written Exam 1 (WE1)

September 3rd, 2021

Last Name
First Name
Id number (Matricola)

Notes

1. The exam is not valid if you do not fill in the above data.
2. Write your answers on these pages. Extra sheets will be ignored. You may use a pencil.
3. Incomprehensible handwriting is equivalent to not providing an answer.
4. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
5. You cannot keep a copy of the exam when you leave the room.
6. The exam is composed of three exercises. Read carefully all points in the text!
7. **Total available time: 1h and 30 mins**

Scores of each question:

Question 1 (MAX 8) _____

Question 2 (MAX 4) _____

Question 3 (MAX 4) _____

Question 1 Alloy (8 points)

A grocery store is equipped with a dispenser of barcode readers (let us call them BCRs). Customers who own the grocery store fidelity card can take one BCR at a time. Customers can then use the acquired BCR to read the barcodes of the products they want to buy and add the product to their bill.

Alloy_1 (3 points)
Define suitable Alloy signatures – and any related constraints – to describe the dispenser, BCRs, customers, products and bills.

Alloy_2 (2.5 points)
BCRs can be used by customers only if their batteries are charged. If not, they should be in the dispenser. Model also this aspect in terms of new and/or modified signatures and related constraints.

Alloy_3 (2.5 points)
Model the operation *getBCR* that extracts the reader from the dispenser and associates it with a customer.

Solution

Alloy_1

```
sig FidelityCard {}
sig Customer {
   card: lone FidelityCard,
   bill: lone Bill
}

sig Bill {
  productItems: set ProductItem
}

sig ProductItem {
  product: Product
}

sig Product {
  cost: Int
}{cost > 0}

sig Dispenser {
  stockedBCRs: set BCR
}

sig BCR {
   assignedTo: lone Customer
}{ #assignedTo > 0 implies assignedTo.card != none }

fact noMoreThanOneCustomerPerBCR {
  no disj bcr1, bcr2: BCR | bcr1.assignedTo & bcr2.assignedTo != none
}
```

Alloy_2
We must modify the BCR signature to include the concept of battery status. We add also additional signatures to model the two possible states of the battery and a fact that ensures only charged BCRs can be assigned to customers.

```
abstract sig BatteryStatus {}
one sig Charged extends BatteryStatus {}
one sig NotCharged extends BatteryStatus {}
sig BCR {
   bs: BatteryStatus,
   assignedTo: lone Customer
}{ #assignedTo > 0 implies assignedTo.card != none }

fact customersUseChargedBCRs {
  all bcr: BCR | (bcr.assignedTo != none implies bcr.bs in Charged) and
                 (bcr.bs in NotCharged implies bcr in Dispenser.stockedBCRs)
}
```

Alloy_3

```
pred getBCR (d: Dispenser, d': Dispenser, c: Customer, c': Customer) {
   c.card != none
   BCR.assignedTo & c = none
   some bcr: BCR | bcr in d.stockedBCRs and bcr.bs in Charged
   c'.card = c.card
   c'.bill = c.bill
   c' in BCR.assignedTo
   (assignedTo.c') in d.stockedBCRs
   d'.stockedBCRs = d.stockedBCRs - (assignedTo.c')
}
```

Question 2 Function Points Analysis (4 points)

Referring back to the grocery store example introduced in the previous question, consider the complete information system for the store. Besides the dispenser and the BCRs, it includes some cash registers that can either interface with the BCRs to collect information about the items scanned by the customers, or are manually operated by a cashier. Cash registers print the bills for the user, register the received amount of money and compute the change to be given to the customer. An external and preexisting central system communicates with the cash registers to receive information about the issued bills and the stored amount of money.

Identify the function points associated with this system specifying, for each of them, the corresponding classification in terms of complexity (simple, medium and complex). Provide the rationale for your decisions.

Solution

Internal Logic Files:

BCR (id, batteryLevel, currentCustomer)

Dispenser (id, storedBCRs)

Customer (id, fidelityCard, billID)

Bill (id, productID, numberOfItems)

Product (id, cost)

The dataset has a simple structure. BCRs and Dispensers are limited in number as well, so they are certainly simple. Customers, Bills and Products may include a relatively large amount of data, so their complexity can range from simple to complex.

External Interface Files:
Even though the system interfaces with the external server, from the description it does not appear to receive data from it. Rather, it communicates data to this central server. So, we do not have EIF.

External Input:
release BCR: this operation is offered by the Dispenser. It is certainly a simple one.
scan product: this operation is offered by the BCR. It is simple.
readBCR: this operation is offered by the cash register. It is simple.
readProductCode: this operation is offered by the cash register. It is simple.
registerAmount: this operation is offered by the cash register. It is simple.

External Inquiries:
check battery level: this operation is offered by the BCR. It is simple.
computeChange: this operation is offered by the cash register. It is simple.

External Outputs:
printBill: this operation is offered by the cash register. It is simple.
sendBillToCentral: this operation is offered by the cash register. It is simple.

Question 3 Symbolic Execution (4 points)

Consider the following function:

```
0  int proc ( int x, int y ) {
1          int n;
2          if ( x==y )
3               return 1;
4          else
5               if ( x>y ) {
6                    y=x*2;
7                    n=0;
8               }
9               else n=1;
10         while ( x<y ) {
11                n=n+1;
12                x=x+2;
13         }
14         return n;
15 }
```

Considering positive values for variables *x* and *y*, answer to the following questions:

SE_1 (1 point)
Identify the paths in the program that enter in the loop *at most* once per path.

SE_2 (3 points)
Derive the path conditions corresponding to the execution of all the identified paths showing how to get to the result you propose.

Solution

SE_1

The possible paths in the program entering in the loop at most once per path are the following:

0, 1, 2, 3
0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 10, 14
0, 1, 2, 4, 5, 9, 10, 11, 12, 13, 10, 14
0, 1, 2, 4, 5, 9, 10, 13, 14
0, 1, 2, 4, 5, 6, 7, 8, 10, 13, 14

SE_2

Symbolic execution for path 0, 1, 2, 3:
0 x = X, y = Y
1
2 X == Y
3 X == Y

The path condition is then x == y

Symbolic execution for path 0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 10, 14:
0 x = X, y = Y
1
2 X != Y
4
5 X > Y
6 y = 2*X
7 n=0
8
10 X < 2*X (this is always true when considering positive values for X and Y)
11 n = 1
12 x = X+2
13
10 X+2 >= 2*X (this is possible if X = 2)
14

The path condition is then x == 2 and y < x

Symbolic execution for path 0, 1, 2, 4, 5, 9, 10, 11, 12, 13, 10, 14
0 x = X, y = Y
1
2 X != Y
4
5 X <= Y
9 n = 1
10 X < Y
11 n = 2
12 x = X+2
13
10 X+2>=Y
14

The path condition is then x < y <= x+2

Symbolic execution for path 0, 1, 2, 4, 5, 9, 10, 13, 14
0 x = X, y = Y
1
2 X != Y
4
5 X <= Y
9 n = 1
10 X >= Y this implies that it must be X == Y which is, however, inconsistent with the path condition that needs to be true at line 2.

There is no path condition that makes this path feasible

Symbolic execution for path 0, 1, 2, 4, 5, 6, 7, 8, 10, 13, 14
0 x = X, y = Y
1
2 X != Y
4
5 X > Y
6 y = 2*X
7 n=0
8
10 X >= 2*X this is not possible since we are assuming that X is positive. Therefore, there is no path condition that makes this path feasible.