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)

**June 24th, 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 7) | ________ |
| Question 2 | (MAX 5) | ________ |
| Question 3 | (MAX 4) | ________ |

**Question 1 Alloy (7 points)**

Consider a transportation company that is organizing a package pick-up service.
The company has a certain number of trucks and drivers. All trucks have the same size. Each day, each driver drives a single truck (the same driver can drive different trucks in different days).
A pick-up request is specified by the origin and the size of the package (for simplicity, assume there are only two sizes, small and large).
A travel plan is associated with a driver and a truck in a certain day and is defined in terms of the packages that must be collected by that driver during the day, using the specified truck.

**Point A (2 points)**
Define suitable Alloy signatures – and any related constraints – to describe the problem above.

**Point B (2 points)**
Define a signature `TruckStatus` that represents the snapshot of a truck, that is, the truck with its current content and current driver. Ensure that the following constraint holds: a truck never exceeds its maximum capacity, which corresponds to an amount of packages whose size is equivalent to 2 large packages. We consider that the size of one large package is equivalent to the size of 4 small packages.

**Point C (1.5 points)**
Formalize the following constraint: a travel plan should never exceed the truck capacity.

**Point D (1.5 points)**
Formalize the following constraint: drivers and trucks cannot be assigned to multiple travel plans in the same day.

**Solution**

**Point A**

```
abstract sig Size {}
one sig Small extends Size {}
one sig Large extends Size {}
sig Place {}
sig Day {}

sig Package {
      origin: Place,
      s: Size
}
/* Note that package in this solution is incorporating also the concept of pickup
request. Another valid option is to make this concept explicit and distinguish
between a package with its size and a pickup request that includes a package and
the information about its origin. In this second solution, the travel plan will
include pickup requests instead of packages */
sig Driver {}

sig Truck {}


sig TravelPlan {
      d: Day,
      o: Driver,
      packages: set Package,
      t: Truck
}
```

**Point B**

```
pred sizePackagesCoherent(cc: set Package) {
      #cc <= 2 or
      (#cc <= 8 and no p: Package | p in cc and p.s = Large) or
      (#cc >2 and #cc <=5 and one p: Package | p in cc and p.s = Large)

}

/* Note the definition of the predicate as a way to simplify the coherency check.
There are two advantages in this solution: 1) we keep the relative complexity of
this check encapsulated in the predicate. TruckStatus is much simpler at this
point. 2) we can reuse this same predicate also in point C */

sig TruckStatus {
      t : Truck,
      currentContent: set Package,
      currentDriver: lone Driver
}{ sizePackagesCoherent[currentContent] }

/* Note that we could include in TruckStatus the corresponding TravelPlan. In this
case, some integrity constraints should be added to ensure that the truck is the
one defined in TravelPlan, the currentContent is a subset of the set of packages
in TravelPlan, the currentDriver is the one in TravelPlan. We did not include this
relation as the text does not explicitly require the truck status to represent the
situation of the truck while taking care of a TravelPlan and it is reasonable to
imagine that a truck could be involved also in other services other than the ones
associated to the definition of a travel plan. */
```

**Point C**

```
fact noTravelPlanExceedsCapacity {
      all tp: TravelPlan | sizePackagesCoherent[tp.packages]
}
```

**Point D**

```
fact driverAndTruckUniqueness {
  no disj tp1, tp2: TravelPlan |
                          tp1.d = tp2.d and ( tp1.o = tp2.o  or  tp1.t = tp2.t )
}
```

## Question 2 Testing (5 points)

Consider the following C function:

```
0 int funct(int x, int y)
1  if (x > 0 && y > 0) {
2     if (y % 2 == 0)
3        return -1;
4     while (x*y % 2 == 0)
5        x = x / 2;
6     while ((x + y) % 2 == 0)
7         x = x  - 1;
8  }
9  return x+y;
```

1. Symbolically execute the program covering the following paths, highlighting the path condition associated with each path:
    1. 0, 1, 2, 4, 5, 4, 5, 4, 6, 7, 6, 7, 6, 8, 9
    2. 0, 1, 2, 4, 6, 8, 9

2. Are all instructions in the code reachable through some path, even one not listed in the text, or not? Motivate your answer.

**Solution**

Path 0, 1, 2, 4, 5, 4, 5, 4, 6, 7, 6, 7, 6, 8, 9

| | |
|---|---|
| 0: x = X, y = Y | PC: true |
| 1: | PC: true |
| 2: | PC: X > 0 and Y > 0 |
| 4: | PC: X > 0 and Y > 0 and Y odd |
| 5: x = X/2 | PC: X > 0 and Y > 0 and Y odd and X even |
| 4: | PC: X > 0 and Y > 0 and Y odd and X even |
| 5: x = (X/2)/2 | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even |
| 4: | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even |
| 6: | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even and X/4 odd |
| 7: x = X/4-1 | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even and X/4 odd and Y+X/4 even (this is trivially true given Y and X/4 are both odd) |
| 6: | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even and X/4 odd and Y+X/4 even (this is trivially true given Y and X/4 both odd) |
| 7: | PC: X > 0 and Y > 0 and Y odd and X even and X/2 even and X/4 odd and Y+X/4-1 even → this is not possible as if Y+X/4 is even then it is not possible to have that the same value minus 1 is still even. |

In conclusion, this path is not possible.

Path 0, 1, 2, 4, 6, 8, 9

| | |
|---|---|
| 0: x = X, y = Y | PC: true |
| 1: | PC: true |
| 2: | PC: X > 0 and Y > 0 |
| 4: | PC: X > 0 and Y > 0 and Y odd |
| 6: | PC: X > 0 and Y > 0 and Y odd and X*Y odd, which means X odd |
| 8: | PC: X > 0 and Y > 0 and Y odd and X odd and X+Y odd → this is not possible as the sum of two odd numbers is always an even number. In conclusion, this path is not possible. |

Even if the two paths above are unfeasible, still all instructions in the function can be reached under certain conditions. For instance, based on the analysis of path 0, 1, 2, 4, 5, 4, 5, 4, 6, 7, 6, 7, 6, 8, 9, which is unfeasible, we can conclude, instead, that it is possible to jump out of the second loop following the path: 0, 1, 2, 4, 5, 4, 5, 4, 6, 7, 6, 8, 9. This one allows all instructions except the one at line 3 to be covered. In turn, instruction 3 can be covered by path 0, 1, 2, 3, for which it is enough that x and y are both positive, and y is even.

**Question 3 Project management (4 points)**

A charity wants to build an information system supporting its processes in delivering humanitarian aid. More specifically, the charity wants to support its employees in: 1) entering and updating information about the assisted people (beneficiaries); 2) entering the amount of funds given to a beneficiary a certain day; 3) inserting and updating the information concerning the donors, including the amount of their donations; 4) searching donors and beneficiaries according to various criteria (name, region, amount of money received, etc.); 5) visualizing aggregated information such as the number of beneficiaries, the type of support they receive, the number of donors, etc.

As the charity was created by grouping together some local organizations, the data concerning the beneficiaries supported directly by these local organizations are stored in other local information systems.

Identify the function points associated with this system specifying, for each of them, the corresponding classification in terms of complexity (Simple, Medium and Complex).

**Solution**

Internal Logic Files:

- Beneficiaries (include name, surname, address, annual income) → simple
- Donors (include name, surname, address) → simple
- Incoming funds (include the date, donor identifier, amount received, description) → simple
- Outgoing funds (include date, beneficiary identifier, amount spent, description) → simple

All these can be considered simple because of their structure and the limited number of entries.

External Interface Files:

We can assume to acquire from the external organizations info about local beneficiaries and donors as well as aggregated information about the performance of the local organization (amount of incoming and outgoing funds, number of assisted people, number of donors, …). All can be considered simple because of their structure and the limited number of entries.

External Input:

- Login/logout → simple
- Insert beneficiary → simple
- Update beneficiary → simple
- Insert donor → simple
- Update donor → simple
- Insert a new entry concerning incoming funds → it requires the join of two tables. Still, it can be considered simple.
- Insert a new entry concerning outgoing funds → it requires the join of two tables. Still, it can be considered simple.

External Inquiries:

- Search for a specific beneficiary → simple
- Search for the support a beneficiary has obtained so far → simple
- Search for a donor → simple
- Search for beneficiaries with certain characteristics (e.g., living in a certain area, with a particularly low income, …) → simple
- Search for donors with certain characteristics → simple

External outputs:

Create a report for the management with information about the number of beneficiaries supported in a timeframe, the donors, the money acquired, the money spent, … → given that this report could be quite articulated, we can assume a medium complexity