NTNU – Trondheim
Norwegian University of
Science and Technology
,

Department of Computer Sciences

Examination paper for TDT 4242
Advanced Software Engineering

Academic contact during examination: Jingyue Li

Phone: 9189 7446

Examination date: May 26. 2017

Examination time: 09:00 AM to 1:00 PM

Permitted examination support material: Code C – Pocket calculators allowed

Other information: Exam developed by Jingyue Li and checked by John Krogstie

Language: English

Number of pages: 7 (including this page and the Appendix 1)

Checked by:

_____

Date Signature

Introduction

In this exam, you can score a maximum of 50 points. The remaining 50 points for the semester comes from the compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

• Document the necessary assumptions

• Explain why you need them

Your answers should be brief and to the point.

Task 1 – Requirements engineering (14 points)

The case is a snow clearing robot. See Appendix 1 for more details

Task 1.1 Requirement quality (6 points)

Try to categorize the textual requirements in Appendix 1 according to the requirements quality metrics: *ambiguity*, *inconsistency*, *forward referencing*, *opacity*, and *noise*. (4 points). If one requirement has several quality issues, list and explain all of them.

Then, try to fix the requirements quality issues of each requirement in the snow clearing robot in Appendix 1 and write down the improved requirements. (2 points)
(Note: There is no single correct answer of improved requirements of the snow clearing robot. You can just formulate improved requirements based on what you think a snow clearing robot should be. The key point here is to show you know how to create high quality requirements).

SG (There are possibly several quality issues in each requirement. The solution below are the minimum requirements quality issues that should be identified and corrected.):

1. The robot is switched on and off with a button on the instrument panel.

   (Ambiguous, says a button, not which button)

2. When the robot identifies that the snow on the ground around itself reaches to a certain depth centimeter, the robot should automatically start and move to clear the snow.

   (forward referencing, says a certain depth, does not specify the number)

3. When the user presses the remote-control key, the user can take over and have a full control of the robot to be able to move it forward, backward, left turn, and right turn. However, the robot can still make its own decision to move or not

   (Inconsistency)

4. When the air temperature is below 0 degree Celsius, the robot should go to the charge site to charge its battery.

(opacity, no clear link between air temperature and battery charge)

5. The robot should be able to identify obstacles and avoid collision with the obstacles.

   (Ambiguity, which type of obstacles, dogs, human, cars?)

6. The robot should be able to turn to a safe mode when it is in an unsafe environment.

   (forward referencing, does not define what a safe mode is and what an unsafe environment mean)

7. The robot should maintain a safe speed when clearing the snow.
   (forward referencing, does not define the value of safe speed)

8. The robot should be able to blink light to another snow clearing robot to say hello when they meet.
   (*noise. It is not relevant requirement*)

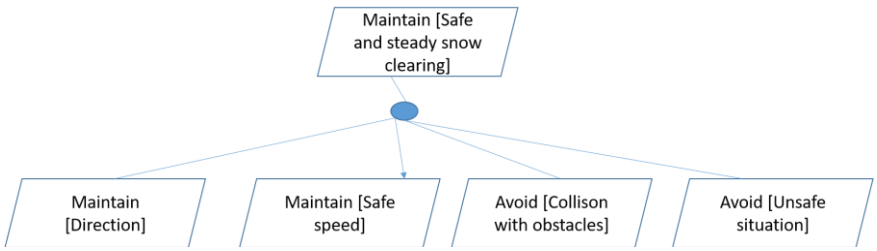
Task 1.2 GORE (4 points)

Based on your improved requirements, use the informal temporal patterns below to describe two upper levels of goal decomposition for the snow clearing robot case in Appendix 1.

Achieve [TargetCondition]
Cease[TargetCondition]
Maintain[GoodCondition]
Avoid[BadCondition]

Improve[TargetCondition]
Increase[TargetQuantity]
Reduce[TargetQuantity]
Maximise[ObjectiveFunction]
Minimise[ObjectiveFunction]

SG (one out of many possible solutions):



Task 1.3 Requirement prioritization (4 points)

Prioritize the following high-level functional requirements of an **Intelligent Traffic Control System (ITCS)** using the cumulative voting method (2 points) and binary priority list method (2 points).
(Note: There is no single correct answer of ranking of the requirements. The key point is to show that you know how to use the two methods to prioritize requirements)

Req1. The ITCS system should be able to log and store traffic data collected through sensors into a server.

Req2: The ITCS system should be able to manage the traffic light to prioritize emergency vehicle.

Req3: The ITCS system should be able to manage the traffic light to decrease the amount of traffic congestions.

Req4: The ITCS system should be able to manage the traffic light to minimize the pollution produced by road traffic.

Req5: The ITCS system should be able to manage the traffic light to minimize pedestrians' waiting time when passing crosses.

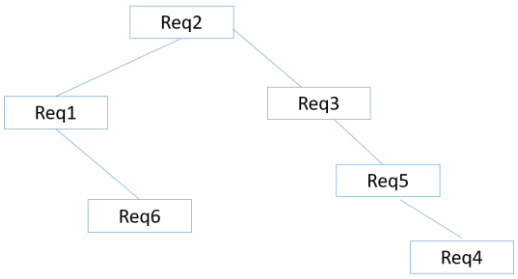
Req6: The ITCS system should be able to analyze the traffic data and make a report.

SG (one out of many possible solutions):

- *Cumulative voting method*

  Req1(8%), Req2(35%), Req3(25%), Req4(12%), Req5(15%), Req6(5%)

- *Binary priority list method*



Task 2 – Testing (15 points)

Task 2.1 Domain testing (3 points)

An online shopping web application has three variables (Availability, Payment method, and Delivery method). The possible values of the variables are as follows.

- Availability: Available (AVA), Not In Stock (NIS), DIScontinued (DIS)
- Payment method: Credit Card (CC), Gift Voucher (GV)
- Delivery method: Mail (MA), UPS (UPS), Fedex (FE)

You task is to write all the 2-way combinatorial test cases (Abbreviations of each variable value can be used)

SG:
If 2-way combinatorial must cover the following pairs (If all pairs are listed correctly, 2 points will be granted)
Availability-Payment method: AVA-CC, AVA-GV, NIS-CC, NIS-GV, DIS-CC, DIS-GV
Availability-Delivery method: AVA-MA, AVA-UPS, AVA-FE, NIS-MA, NIS-UPS NIS-FE, DIS-MA, DIS-UPS, DIS-FE
Payment method- Delivery method: CC-MA; CC-UPS; CC-FE, GV-MA, GV-UPS, GV-FE

9 test cases are needed to cover all the pairs (If the test cases are listed correctly, all 3 points will be granted)

Availability	Delivery	Payment
AVA	MA	CC
AVA	UPS	GV
AVA	FE	CC
NIS	MA	GV
NIS	UPS	CC
NIS	FE	GV
DIS	MA	CC
DIS	UPS	GV
DIS	FE	CC

Task 2.2 Regression testing (6 points)

Suppose we have a code snippet as follows. We initially have a test set T with 4 test cases to test the code. The 4 test cases in the test set T are:

t1: <x = 1, y = 2>
t2 : <x = 1, y = 3>
t3 : <x = 3, y = 1>
t4: < x= 2, y = 2>

If the line “return b*b” in function f1 is changed to “return a*b”, your task is to choose “safe re-testable subset” from T using graph-walk approach with CFG (Control Flow Graph) and explain in details how you choose the subset.

```
int  M (int x, int y){
  int z;
  if (x>=y)
     z = f1 (x, y);
  else
     z = f2(x, y);
  return z;
}
```

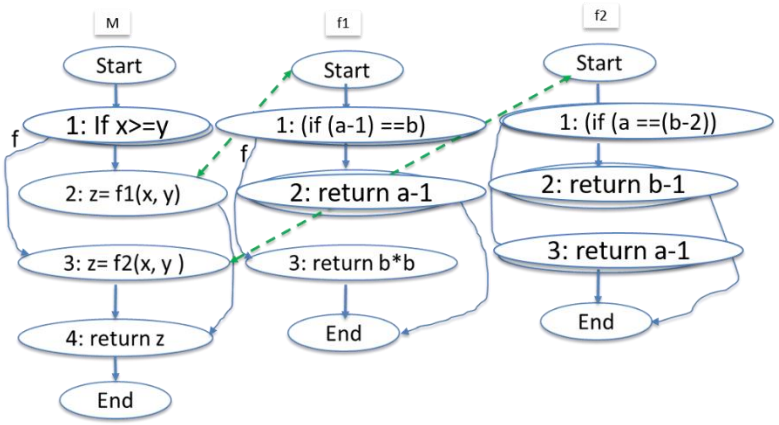
```
int f1( int a, int b){
  if ((a-1) == b)
     return a-1;
  else
     return b*b;
}
```

```
int f2( int a, int b){
  if (a == (b-2))
     return b-1;
  else
     return a-1;
}
```

Code snippet

SG:

- Step 1: Establish trace between test case and CFG (control flow graph) nodes



Test set T
t1: <x = 1, y = 2>
t2 : <x = 1, y = 3>
t3 : <x = 3, y = 1>
t4: < x= 2, y = 2>

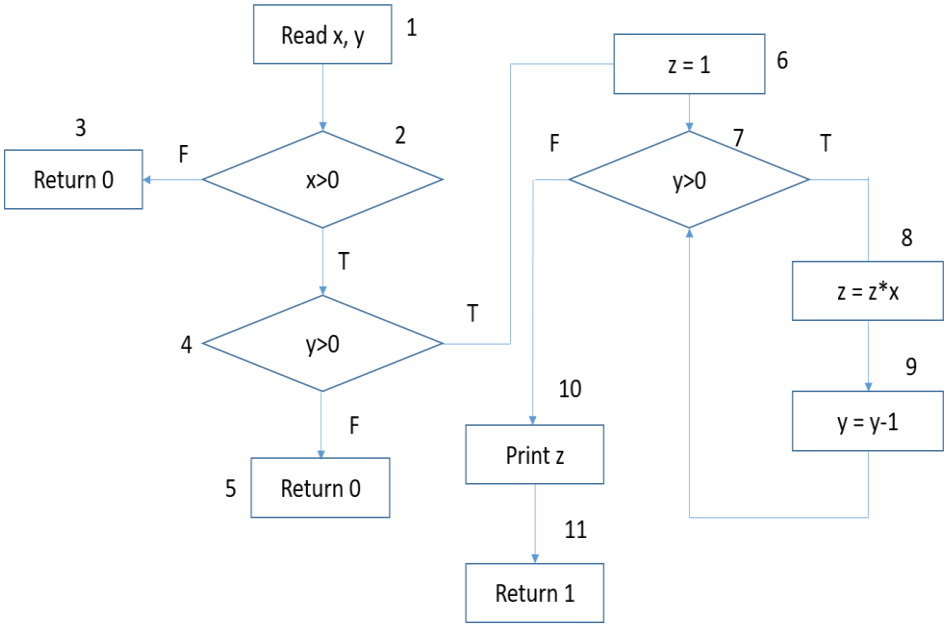
Functions	Nodes			
	1	2	3	4
M	t1, t2, t3 t4	t3, t4	t1, t2	t1, t2, t3, t4
f1	t3, t4	-	t3, t4	-
f2	t1, t2	t2	t1	-

- Step 2: Compare P and P' to find differences
  Node 3 of f1 has been changed.

- Step 3: Select test cases from T that traverse the changed CFG nodes
  Test cases t3 and t4 will be chosen for regression testing, because they traverse the changed CFG node

Task 2.3 Dataflow testing and test coverage (6 points)

For the following CFG (Control Flow Graph), find out paths to be covered for variables x, y, z, if All Uses (AU) is the test strategy to be applied.



SG (The paths must cover the following paths, although the paths can be much longer ones):
(1, 2, x)
(1, 8, x)
(1, 4, y)
(1, 7, y)
(1, 9, y)
(9, 7, y)
(9, 9, y)
(6, 8, z)
(6, 10, z)
(8, 8, z)
(8, 10, z)

Task 3 – Advanced topics (15 points)

Task 3.1 (3 points)

Explain the three high level categories of OSS license types and give one example license in each category.

SG:

- *Strong-copyleft (e.g. General Public License (GPL))*
  - *Once software is licensed by a developer, the **derivative** work (includes major copyright-protected elements of an original) must be licensed similarly*
- *Weak-copyleft (e.g. Lesser General Public License (LGPL))*
  - *Allows developers integrate software released under the LGPL into their own (even proprietary) software without being required to release the source code of their own components*
  - *Primarily used for libraries*
- *Non-copyleft (e.g. Berkeley Software Definition (BSD) and MIT license)*
  - *No obligation to inherit the original license*

Task 3.2 (3 points)

Explain what "operating system-centric" software ecosystem is (1 point) and list its success factors (1 point) and challenges (1 point).

SG:

Characteristics

- *Domain independent*
- *Installed for every device*
- *Focused on stand-alone applications*
- *Focus on development tools for developers*

Success factors

- *Number and relevance of applications built*
- *OS needs to evolve constantly*
- *Number of customers*

Challenges

- *Variation in underlying hardware configurations*

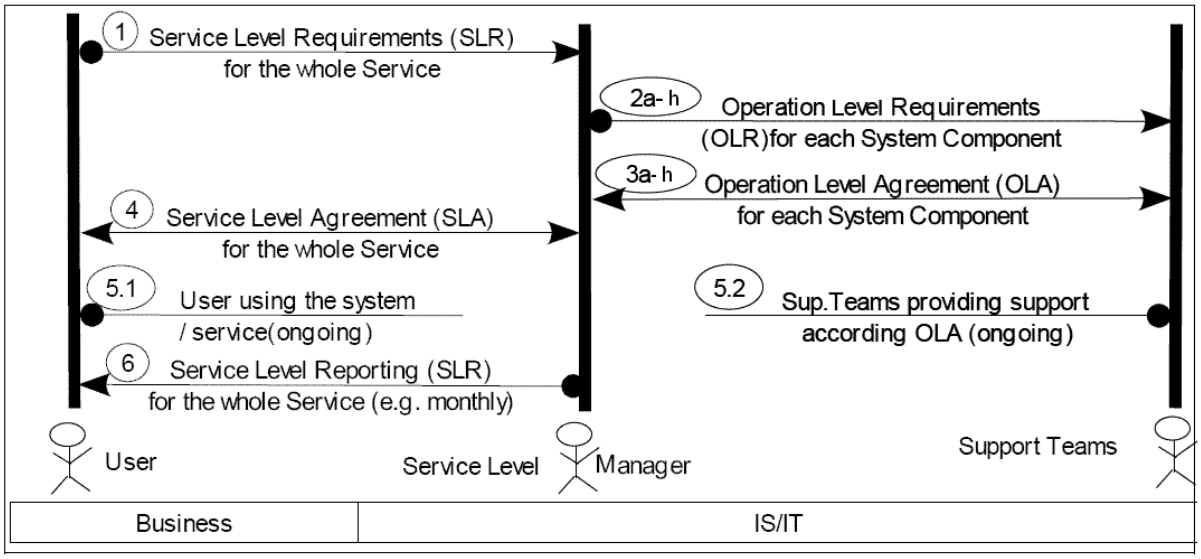
Task 3.3 (4 points)

Explain SLA (Service level agreement) and its content (1 points), OLA (Operations level agreement) and its content (1 points), and the relationship between SLA and OLA (2 points).

SG:

- ***SLA** is the overall agreement between IS/IT and the business department, usually includes*
  - *Services being made available*
  - *Service performance*
  - *Cost to provide the service*
  - *Service provider and customer responsibility*

  - *Performance monitoring and report*
  - *Disaster recovery process*
  - *Periodic review process*
- ***OLAs** are agreements between different IS/IT-departments and the Service Level Manager*
- *Relationship between SLA and OLA (A brief explanation of the idea of the chart is ok)*



Task 3.4 (5 points)

Explain why systematic reading techniques outperforms unsystematic reading techniques (4 points), and list and explain one systematic reading technique (1 point).

SG:
Unsystematic reading, where one or more readers scan through the document and look for defects, which are shaded as vertical, horizontal, or slanted lines. There are at least two problems with that approach owning to its unsystematic nature: (a) there are overlaps on the portions covered by different readers, and (b) there are regions that are not covered by any reader.

Systematic reading, where different readers are purposely selected to exam the software artifact, based on their individual expertise. They are also given specific instructions on where and how to detect defects. These two ensure that there will be no or minimal overlaps among what they will cover and the entire document is covered. Systematic approaches with specific responsibilities improve coordination and reduce gaps, which increases the overall defect detection effectiveness of review.

Systematic reading techniques: Defect-based reading, perspective based reading.

Task 4 – Multiple choice (6 points, one point for each question. You get the one point if all correct answers are chosen)

1) What are the desirable properties of the component-based architecture?

a) Parameterizable components
b) Components should not be customizable
c) Supports component development in multiple programming languages
d) Allows easy distribution of components from seller to buyer

SG: a, c, d

2) Which are the challenges of service oriented software engineering?
   a) Dynamic software evolution
   b) Single point of failure
   c) Complex negotiation process
   d) Hard to understand when automated

SG: b, c, d

3) What are benefits of TDD (Test-Driven Development)?
   a) TDD is always applicable
   b) TDD can lead to higher code coverage
   c) TDD will guarantee a shorter development time
   d) TDD helps simplify debugging

SG: b, d

4) What are differences between Model-Based Testing (MBT) and traditional manual testing?
   a) For MBT, test cases are tightly coupled to the model
   b) For MBT, test cases are generated automatically from models
   c) For MBT, there is a high cost in early phases of the project
   d) For MBT, test oracles still need to be input manually

SG: a, b, c

5) What can cause project cost/effort derives from its original cost/effort estimation?
   a) Pressure from managers to give low bid to get the project
   b) Over-optimistic estimates
   c) Frequent major requirement changes
   d) Major changes in design and implementation

SG: a, b, c, d

6) Which are advantages of scenario-based testing?
   a) Easy to reuse
   b) It is a user focused testing
   c) Does not require working features before testing
   d) Can expose requirement related issues

SG: b, d

Appendix 1 – Snow clearing robot

(Note: This picture is just to illustrate what a snow clearing robot is. This robot is not necessary linked to the requirements below)

Requirements:

Req1: The robot is switched on and off with a button on the instrument panel.

Req2: When the robot identifies that the snow on the ground around itself reaches a certain depth, the robot should automatically start and move to clear the snow.

Req3: When the user presses the remote-control key, the user can take over and have a full control of the robot to be able to move it forward, move it backward, make it turn left, and make it turn right. However, the robot can still make its own decision to move or not.

Req4: When the air temperature is below 0 degree Celsius, the robot should automatically go to the charge site to charge its battery.

Req5: The robot should be able to identify obstacles and avoid collision with the obstacles.

Req6: The robot should be able to turn to a safe mode when it is in an unsafe environment.

Req7: The robot should maintain a safe speed when clearing the snow.

Req8. The robot should be able to blink the light to another snow clearing robot to say hello when they meet.

Cover page

Department of (department): Department of Computer Science

Examination paper for (course code) (course title): (TDT4242) (Advanced Software Engineering)

Academic contact during examination: Jingyue Li

Phone: 9189 7446

Examination date: 7 June 2018

Examination time (from-to): 9.00 – 13.00

Permitted examination support material: C

Other information:

Students will find the examination results on the Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

Introduction

In this exam, you can score a maximum of 50 points. The remaining 50 points for the semester comes from the compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

1. Requirements prioritization

(2 points)

Explain how to prioritize software requirements using the "cumulative voting" approach, and benefits and limitations of this approach.

Answers:

(1/2 points) Stakeholders are given many imaginary units (100 dollars, 1000 points, etc.). Points are distributed among requirements to prioritize

- 100 points

  Req1: 40

  Req2: 20

Req3: 30

Req4: 10

(1/ 2 points) Benefits and limitations

- Easy to use and fast
- Eligible to small number of requirements, not scalable

2. Requirements case study

(16 points)

Company A is going to development a robot to deliver objects, e.g. mail, to employees' office. The requirements that Company A receive from its customer are as follows:

We are going to develop a robot to help deliver objects, e.g. mail, to office to our employees. We want a robot that is efficient, safe, and energy saving. We hope that after we give a map to the robot, and mark out the office that we want to the robot to go, the robot can reach there autonomously. In addition, we would like the robot to have sensors to avoid collisions with obstacles, because there are always people walking around in the office area, and people can also leave something on the floor. The robot should move autonomously without our intervention. However, we sometimes want to interrupt the robot if necessary. We want the robot to move with a default speed. The robot should leave the object outside the door of the office. We hope that the robot use battery to save energy and can charge by itself. When the robot is close to people, the robot should go to the charge site to charge its battery.

Task 1: Make requirement boilerplates and re-express the requirements using the boilerplates you make (2 points)

Task 2: Categorize the textual requirements according to the requirements quality metrics: *ambiguity*, *inconsistency*, *forward referencing*, and *opacity*. If one requirement has several quality issues, list and explain all of them. Then, try to fix the requirements quality issues of each requirement and write down the improved requirements. (6 points)

Task 3: Make a requirement reading guideline for testers, using template for perspective-based reading.

The **Introduction** section of the guideline is "Requirements must be testable". Your task is to fill in the **Instruction** and **Questions** sections of the guideline. In the **Questions** section, you need to fill in at least 3 questions. (4 points)

Task 4: Based on your improved requirements, use the informal temporal patterns below to describe goal decomposition and agent responsibility model of the robot to deliver objects. In the agent responsibility model, a goal should be placed under the responsibility of one or several agents. If the agent is not specified in the customer requirements, you can define an agent yourself.

(Note: If it is difficult to draw the goal decomposition and agent responsibility model in the digital exam context, you can use text to describe them. For example, top level goal **1** is: ... The sub-goal 1.1 is ..., the sub-goal 1.2 is ..., 1.1. and 1.2 have AND relationship, the agent of 1.1 is ...) (4 points)

Achieve [TargetCondition]
Cease[TargetCondition]
Maintain[GoodCondition]
Avoid[BadCondition]

Improve[TargetCondition]
Increase[TargetQuantity]
Reduce[TargetQuantity]
Maximise[ObjectiveFunction]
Minimise[ObjectiveFunction]

To answer the questions in a structured manner, it is better to answer like follows:

Task 1: ...

Task 2: ...

Task 3: ...

Task 4: ...

Answers:

Task 1: (1/2 points) You need to show meaningful boilerplates. Examples are:

The <system> shall <function>

The <user> shall able to <capability>

(1/2 points) You need to show that the requirements are reformulated using your boilerplates.

Task 2: There are more than one case in each requirement error category. These are just examples. If you give other valid cases, you can also get full points.

(1/6 points) ambiguity: “we would like the robot to have sensors to avoid collisions with obstacles”, no definition of obstacles.

(1/6 points) inconsistency: “The robot should move autonomously without our intervention. However, we sometimes want to interrupt the robot if necessary.” Inconsistent control.

(1/6 points) forward referencing: “We want the robot to move with a default speed.” No definition of default speed.

(1/6 points) opacity: When the robot is close to people, the robot should go to the charge site to charge its battery. Requirement rational or dependencies are invisible, or are not meaningful.

(2/6 points) Meaningful fix of the above requirements errors. Each fix counts 0.5 point.

Task 3: (1/4 points) Instruction: Read the requirement to answer the following questions.

(3/4 points) Questions (These are many valid questions. These are just some examples. If you provide minimum three valid questions, you will get full points):

- Are there indefinite pronouns in the requirements?
- Is there passive voice in the requirements?
- Are there vague words in the requirements?
- Is it costly to test the requirement?

Task 4: Several answers could be correct here. The goal-model should contain AND&OR relationship (1 point), agents are linked to goals (1 point), meaningful goal model overall (2 points).

3. DevOps terms

(3 points)

Explain the following three DevOps terms: lead time, cycle time, and percentage complete and accurate.

Answers:

(1/3 points) Lead time is the time from the moment when the request was made by a client and placed on a board (or a ticket is created) to when all the work on this item is completed and the request was delivered to the client. It is the total time the client is waiting for an item to be delivered.

(1/3 points) Cycle time is the amount of time, that the team spent actually working on this item (without the time that the task spent waiting on the board or the ticket). The cycle time should start being measured, when the item task enters the “working” column, not earlier.

(1/3 points) Percentage complete and accurate: percentage of artifacts that could be consumed by the practitioners receiving them without need to modification or rework.

4. Firewall approach

(2 points)

Explain what the firewall approach of regression test selection is, and explain how to determine the firewall of object-oriented systems.

Answers:

(1/2 points) A firewall in regression testing separates the classes that depend on the class that is changed from the rest of the classes. Firewall approach is based on the first-level dependencies of modified components.

(1/2 points) Given two successive versions of an OO-system, find the difference of the two versions and identify those classes that have changed. If any of the changed classes are in an inheritance hierarchy, also consider descendants of the changed class as changed. For each changed class, identify all the classes that send messages to the changed class or receive messages from the changed class and include them in the "firewall".

5. Domain testing

(5 points)

An online PC shopping web application has five variables (Availability, Payment method, Delivery method, PC type, and Damage insurance).

The possible values of the variables are as follows.

- Availability: Available (AVA), Not In Stock (NIS), DIScontinued (DIS)
- Payment method: Credit Card (CC), Gift Voucher (GV)
- Delivery method: Mail (MA), UPS (UPS), Fedex (FE)
- PC type: DESKtop (DESK) and LAPtop (LAP)
- Damage insurance included in price: With Insurance (WithIN), Without insurance (NoIN)

You task is to write all all-pair combinatorial test cases (Abbreviations of each variable value can be used)

Answers:

(5/5 points) Several combinations can be correct. This is just one possible solution. All pairs mean that "every value of a variable is combined with every value of every other variable at least once". If you miss any pairs, or if you do not show that you understand how to make the all-pair test cases, you will lose all points.

Availability	Delivery method	Payment method	PC type	Insurance
AVA	MA	CC	DESK	WithIN
AVA	UPS	GV	LAP	NoIN
AVA	FE	CC	DESK	WithIN
NIS	MA	GV	LAP	NoIN
NIS	UPS	CC	**LAP**	WithIN
NIS	FE	GV	**DESK**	NoIN

DIS	MA	CC	DESK	**NoIN**
DIS	UPS	GV	LAP	**WithIN**
DIS	FE	CC	DESK	WithIN

6. MBT

(2 points)

Explain what MBT (Model-based testing) is, and how MBT works.

Answers:

(1/2 points) What MBT is: MBT is a testing approach where test cases are automatically generated from models. The models are the expected behavior of the system under test and can be used to represent the test strategy.

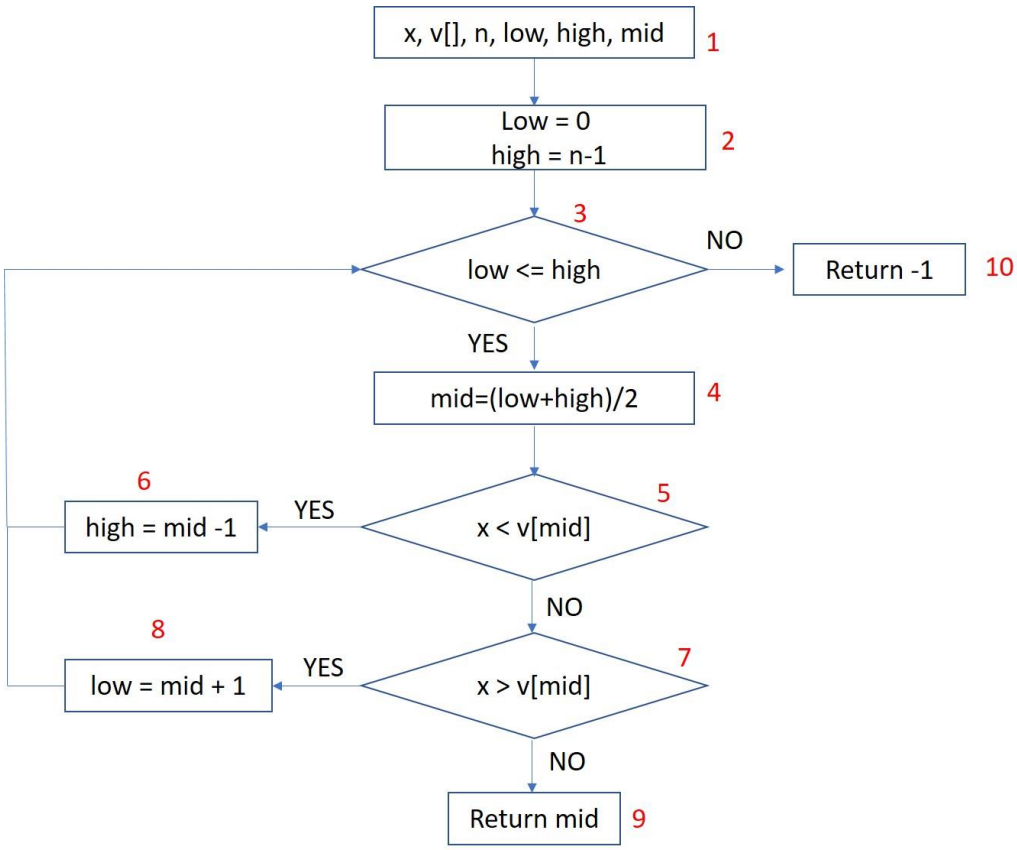
(1/2 points) How MBT works: MBT usually includes the following steps: 1. Design a test model. 2: Select some test generation criteria. For example, the test generation criteria could be full state coverage, if the model is a finite state machine. 3. Generate the tests. 4. Execute the tests.

7. Data flow testing

(4 points)

The following CFG (Control Flow Graph) is from the binary search code. Based on the CFG, your tasks are:

- Identify paths to be covered for the variables "high" and "mid", if All Uses (AU) is the test strategy to be applied




```
int binarysearch(int x, int v[], int n){
        int low, high, mid;
        low = 0;
        high = n-1;
        while (low <= high){
           mid = (low + high) /2;
           if (x < v[mid])
                 high = mid -1;
           else if (x > v[mid])
                 low = mid + 1;
                 else
                   return mid;
        }
        return -1;
}
```

Binary search is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array; if they are unequal, the half in which the target cannot lie is eliminated and the

search continues on the remaining half until it is successful. If the search ends with the remaining half being empty, the target is not in the array.

Answers:

(2/4 points) Paths to be covered for the variables "high".

<2,3><2,4><6,3><6,4>

(2/4 points) Paths to be covered for the variables "mid".

<4, 5><4, 6><4, 7><4, 8><4, 9>

8. Software Ecosystem

(5 points)

Explain characteristics, success factors, and challenges of an application-centric software Ecosystem, and list two examples and explain the examples briefly.

Answers:

For the characteristics and success factors, and challenges, you need list at least two out of the possible items to get full points.

(1/5 points) Characteristics:

- Starts from a successful application
- Provide APIs for developing customer-specific functionality
- Application developers extend the basic application
- Deep integration for data, workflow and user experience

(1/5 points) Success factors:

- Number of customers
- Simplified contribution of 3rd party extensions
- Mechanisms for extending data models and workflows

(1/5 points) Challenges:

- Product versus platform strategy
- Viable business model for 3rd party developers

These examples are just reference examples. If you provide other valid examples, you can also get full points.

(1/5 points) Example 1: Facebook which provides APIs for 3rd party developers to develop customer-specific functionalities.

(1/5 points) Example 2: Amazon also provides APIs for 3rd party developers to develop customer-specific functionalities.

9. Service oriented architecture

(5 points)
List benefits of SOAP (Simple Object Access Protocol) web service and benefits of REST (REpresentational State Transfer) web service?

Answers:

(3/5 points) You need to list minimum 3 items below to get full points

SOAP is good for

- Enterprise services
- High reliability and security
- Asynchronous processing
- Contract-first development
- Stateful operation
- Standards support
- Tool support

(2/5 points) You need to list minimum 2 items below to get full points

REST is good for

- Limited bandwidth (smaller message size)
- Limited resources (no XML parsing required)
- Exposing data over internet
- Combing content from many different sources in a web browser

10. Agile distributed development

(6 points)

Explain communication, trust, and control challenges of agile distributed development and list/propose strategies to address those challenges.

Answers:

(1/6 points) Communication challenge: Agile requires more informal communication. However, distributed development makes it difficult to communicate frequently.

(1/6 points) How to address communication challenge (You need to list at least one item):

- Documenting requirements at different levels of formality
- Informal communication but through formal channels
- Constant communication

(1/6 points) Trust challenge: In distributed development, people do not know each other very well. Thus, they do not trust each other very much. However agile development requires high team cohesion and trust.

(1/6 points) How to address trust challenge (You need to list at least one item):

- Focus on well-understood functionalities in early iterations
- Trust but verify

(1/6 points) Control challenge: In distributed development, it is difficult to control process and quality across distributed teams. Thus, frequent quality requirements changes are not welcome. However, in agile development, requirement changes are more acceptable.

(1/6 points) How to address control challenge: Short cycle but not-boxed development

Cover page

Department of (department): Department of Computer Science

Examination paper for (course code) (course title): (TDT4242) (Advanced Software Engineering)

Academic contact during examination: Jingyue Li
Phone: 9189 7446
Examination date: 23 May 2019
Examination time (from-to): 9.00 – 13.00
Permitted examination support material: C
Other information:

Students will find the examination results on the Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

Introduction

In this exam, you can score a maximum of 50 points. The remaining 50 points for the semester comes from the compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

1. Problem 1

(30 points in total)

1) Explain the purposes of establishing forward and backward traceability in software projects and how to establish forward and backward traceability in software projects and artifacts. (5 points)
   Purpose
   - Forward traceability (2 points)
     - Ensure all requirements are implemented
     - Change impact analysis
   - Backward traceability (2 points)
     - Avoid "gold plating"
     - Change impact analysis
     - Defect impact analysis
     - Root cause analysis of defects

   How to (1 point)

   - Traceability matrix
   - Trace tagging

2) Explain what static backward slicing is and how to create backward slicing using data flow information. (2 points)

   The purpose of code slicing is to choose only a subset of code that is relevant to a particular variable at a certain point to minimize the code for more cost-efficient testing, debugging, or testing. (0.5 points)
   Static backward slicing: A slice with respect to a variable *v* at a certain point *p* in the program is the set of statements that **contributes** to the value of the variable v at p (0.5 points)
   To establish static backward slicing, all Defs and All P-uses of a variable v before a certain point p are chosen (1 point).

3) Explain decomposition-based, call-graph based, and path-based integration testing strategies and compare their pros and cons. (6 points)
   Decomposition-based (2 points): integration testing can be a top-down, bottom-up, sandwich, or big bang. The basic idea is to use "stub" and "driver" to simulate the called function or call function.
   - Pros: Intuitive and easy fault isolation
   - Cons: Need "stub" or "driver"

   Call-graph based (2 points): integration testing is based on the call graph of neighbors or interface matrix
   - Pros: Do not need "stub" or "driver"
   - Cons: Difficult to isolate and locate the fault

Path-based (2 points): integration testing is like an enlarged path-based unit test. Here the node is not a statement but a component.

- Pros: Closely coupled with actual system behavior
- Cons: extra effort is needed to identify message path

4) Explain the general regression test selection process. (3 points)
Regression test selection process usually includes three main steps:
    - Run the tests and establish the link between the test and the code executed (1 point)
    - After the code is changed, identify the changed code (1 point)
    - Based on the link between the code and the test, choose only those tests that are related to the changed code and re-run them (1 point)

5) Explain the essential ideas of the two types of standards to verify safety-critical systems. (2 points)
    - "How-standards" focus on how we should work – e.g., how we should work to achieve safety or security. At present, most standards are "How"-standards (1 point)
    - "What-standards" (also called Goal-based standards) focus on what shall be achieved, not on how to achieve it. It is up to the developers to convince the assessor that what they do will meet the standard's requirements (1 point)

6) Explain what adaptive random test is and its benefits. (3 points)

The purpose of the adaptive test is to generate test cases that are evenly spread over the entire input domain as possible. You start with one test case that is generated randomly. Then for the next test case, you pick the one that is the farthest away from all executed test cases. The benefit is that it is good for F-measure, i.e., it usually requires fewer test cases to detect the first failure comparing to random test.

7) Explain why code inspection and testing are complementary. (4 points)

The benefits of testing are that (1 point)

- Can, at least partly, be automated
- Can consistently repeat several actions
- Fast and high volume

However, testing has also limitations (1 point)

- Is only a spot check
- May need several stubs and drivers

The benefits of code inspection is that (1 point)

- Can see the complete picture, not only a spot check
- Can use all information in the team
- Can be innovative and inductive

However, code inspection could be: (1 point)

- Unreliable (people can get tired)
- Slow and low volume

8) Explain MySQL dual licensing. (2 points)
   - MySQL free use (1 point)
     - Within a web site
     - ISP may make MySQL available to its customers for free
     - All projects that themselves run under the GPL
   - MySQL commercial license, e.g., (1 point)
     - Develop and sell a commercial product that is geared toward MySQL as the database

9) Based on the content of the guest lecture "coordination in large-scale agile development," describe some coordination challenges and possible solutions in large scale agile projects. (3 points)
   - Challenges: Scrum-of-Scrum meetings involving representatives from all teams were severely challenged: the audience was too wide to keep everybody interested. Inter-group coordination becomes a major challenge when groups enjoy high levels of autonomy (2 points).
   - Possible solutions (List of anyone below can get 1 point)
     - Have enough formal arenas to enable informed informal coordination
     - Be prepared to add or change the content of the formal arenas over time as needs change
     - Specialized roles in the team contribute to cross-team coordination
     - Mini demos enable faster feedback and closed-loop communication with customers
     - Trust in line with clear decision-making arenas enables autonomous handling of problems with feedback loops to the project

2. Problem 2: Requirement Engineering

(8 points)

Company A is going to pay Company B for developing Autonomous Truck Platooning. Truck platooning (as shown in the above picture) is the linking of two or more trucks in convoy, using connectivity technology and automated driving support systems. These vehicles automatically maintain a set, close distance between each other when they are connected for certain parts of a journey, for instance on motorways.

The requirements Company B got from Company A are as follows.

We are going to develop Autonomous Truck Platooning. We want some numbers of the trucks can drive autonomously, and other trucks are driven by humans. The autonomous trucks should drive at a default speed. We hope the truck platooning can drive along the route we set in GPS before the journey and follow the human drivers' command to change the route. We hope the autonomous trucks should have three chairs for the drivers. We want the truck to drive safely. So the trucks should have sensors to avoid collisions with obstacles because there are always other vehicles or obstacles in the motorway. When there are obstacles on the road, the trucks should stop. We also want the trucks to drive efficiently, which means that the trucks should minimize the distance between them and maximize the speed of the whole Platooning. The trucks should use sensors to detect the distances between trucks and the speed of the trucks.

Task 1: Identify quality issues in these requirements according to the requirements quality metrics: *ambiguity*, *inconsistency*, *forward referencing*, and *opacity*. If one requirement has several quality issues, list all of them. Then, try to fix the requirements quality issues of each requirement and write down the improved requirements. (4 points)

Each category of quality issues can have several cases. If you list any of the cases in the four categories and fix them, you will get one point of each category.

- Ambiguity (1 point): *So the trucks should have sensors to avoid collisions with obstacles because there are always other vehicles or obstacles in the motorway (Not explain obstacles).*
- Inconsistency (1 point): E.g., *We hope the truck platooning can drive along the route we set in GPS before the journey and follow the human drivers' command to change the route (Conflict on who can make the control decision).*
- Forward referencing (1 point): E.g., *The autonomous trucks should drive at a default speed (Does not define default speed).*
- Opacity (1 point): We hope the autonomous trucks should have three chairs for the drivers (not relevant to the autonomous truck).

Task 2: Make requirement boilerplates and re-express the requirements using the boilerplates you make. (2 points)

- The list of meaningful boilerplate is 1 point.
- Re-expressing the requirement using boilerplates is 1 point.

Task 3: Based on your improved requirements, use the informal temporal patterns below to describe goal decomposition and agent responsibility model of the Autonomous Truck Platooning. In the agent responsibility model, a goal should be placed under the responsibility of one or several agents. If the agent is not specified in the customer requirements, you can define an agent yourself. (Note: If it is difficult to draw the goal decomposition and agent responsibility model in the digital exam context, you can use text to describe them. For example, top-level goal **1** is: ... The sub-goal 1.1 is ..., the sub-goal 1.2 is ..., 1.1. and 1.2 have AND relationship, the agent of 1.1 is ...). (2 points)

Achieve [TargetCondition]
Cease[TargetCondition]
Maintain[GoodCondition]
Avoid[BadCondition]

Improve[TargetCondition]
Increase[TargetQuantity]
Reduce[TargetQuantity]
Maximise[ObjectiveFunction]
Minimise[ObjectiveFunction]

To answer the questions in a structured manner, it is better to answer like follows:

Task 1: …

Task 2: …

Task 3: …

This is just one possible solution. Other solutions which can show that the student understands the goal-oriented approach can also get full points. The goal hierarchy and proper use of the temporal

pattern will count 1 point, and the proper link between the goal (AND or OR relation is specified) and agent listed and linked to the goals will count 1 point.

1. Achieve[Safe and Effective driving]

1.1 Avoid[Collisions]

1.1.1 Maintain [More than the minimum safe distance between trucks] – Agents: Camera and Lidar

AND

1.1.2 Cease [Unavoidable obstacle is identied] – Agent: Camera, Lidar

AND

1.1.3 Avoid [Overspeed] – Agent: Speed meter

1.2 Achieve[Effective driving]

1.2.1 Maintain [Speed close to the speed limit] – Agent: Speed meter

OR

1.2.2 Avoid [Uncessary slow down]- Agent: Speed meter, Camera, Lidar

3. Problem 3: Testing

1. Combinatorial test (4 points)

An online car renting web application calculates the car rental prices based on four variables (Party size, Car specification, Mileage/Kilometres, and Damage insurance).

The possible values of the variables are as follows.

- Party size: Small (S), Medium (M), and Large (L)
- Car specification: Air Conditioning (AC), Automatic Transmission (Auto), and Manual Gearbox (Man)
- Mileage/Kilometres: Limited (Lim), Unlimited (Unlim)
- Damage insurance included in the price: With Insurance (WithIN), Without insurance (NoIN)

You task is to write all all-pair combinatorial test cases based on these four variables (Abbreviations of each variable value can be used)

This is one reference solution (4 points).

Party size	**Car specification**	**Mileage/Kilometres**	**Damage insurance**
S	AC	Lim	WithIN
S	Auto	Unlim	NoIN
S	Man	Lim	WithIN
M	Ac	Unlim	NoIN
M	Auto	Lim	WithIN
M	Man	Unlim	NoIn
L	AC	Lim	**NoIN**
L	Auto	Unlim	**WithIN**
L	Man	*Lim*	*WithIn*

2. System test (3 points)

You are assigned a task to run scalability testing and stress testing of a popular online air ticket booking application. Explain the purpose of scalability testing and stress testing and how to perform scalability testing and stress testing to test the application. (3 points)

The purpose of scalability testing is to test how well the system can be scaled up to process the increased workload. To perform scalability testing, the testers can gradually increase the workload to the system, and also add more servers and workload managers, in order to measure how well the system can process the increased workload through adding more servers (1 point).

The purpose of stress testing is to test how well the system can perform acceptably under worse-case condition. The system is deliberately stressed by pushing it to and beyond its specific limits for a while in order to identify some bugs related to memory, e.g., memory leak (1 point).

For the online air ticket booking application, to run scalability testing, the testers will simulate the increased number of parallel transactions and add more servers and configure the workload manager to check how well the added servers deal the increased workload. To run the stress testing, the testers will simulate the maximum number of parallel transactions for a while to see if the performance of the system is still acceptable (1 point).

4. Problem 4: Code refactoring case study

Identify bad code smells in the following code and propose how to refactor them. The proposal of code refactoring could be refactored python code or pseudo-code to explain how to refactor. (5 points)

Note: The code is a working code. Your task is not to identify the security vulnerabilities of the code. Your task is to identify bad code smell and to refactor the code.

```
"""
Package: utils.config
"""

CONFIG_NAME = {
   "ENABLE_LOGGING": "enable_logging",
   "LOGGING_LEVEL": "logging_level",
}

def get_logging_level():
   pass

class ConfigHelper:
   def get(self, config_name, default=None):
      pass

def set(self, config_name, value):
      pass

def _get_settings_helper(self):
      pass

def get_logging_level():
      pass

def is_logging_enabled():
      pass

class LOGGING_LEVEL:
   VERBOSE = "verbose"
   STANDARD = "standard"
```

The main code smells are:

- Duplicated code: def get_logging_level() appears twice (1 point)
- Naming convention: function names or variable names are not inline with Python naming convention (2 points)

- Too small classes or constants : the classes and constants can be moved to a separate file, be removed, or be merged to optimize the readability of the code (2 points)

Introduction

In this course, the written exam will count 50% of the final grade, and the remaining 50% of the final grade comes from the compulsory exercises.

So, your final grade of this course will be:

Points you get from this written exam + points you get from the compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

You need to put all your answers to a .pdf file and upload the .pdf file to Inspera before the examination time expires.

Problem 1 – Use case (5 points)

Many ATM (Automated Teller Machine) of banks allow users to withdraw cash from a bank account and to store money to a bank account. **Based on the use case template taught in this course, write a use case** "store cash to a bank account."

Storing cash to a bank account means that:

- The users can put some money in the ATM
- The ATM can identify whether the money is true/faked
- The ATM can sum up the total amount of the money
- The ATM can add the total amount of the money to the user's bank account

Problem 2 – Goal-oriented requirement engineering (5 points)

Based on goal-models taught in this course, make a goal-model of the control system of the “autonomous bus” using soft goal types. The goal-model should have at least three layers.

(Note: You can use whatever software to draw the goal-model. Handwriting drawing is not acceptable).

Problem 3 – System test (12 points)

Suppose you will lead the system test of a web application that allows users to buy/sell stock shares in real-time. Your task is to explain to your top-level managers the purpose of each type of the following system test and give them at least one test case example.

- Robustness test
- Scalability test
- Stress test
- Load and stability test
- User acceptance test
- Operational acceptance test

Problem 4 – Code review and testing (8 points)

Use the bugs and code smells your group identified in the exercises as examples to explain the pros and cons of testing and code review, and explain why testing and code review complement each other to assure software quality. You need to list at least two bugs your group identified through testing, and at least two code smells your group identified through code review as examples.

Problem 5 – Bug tracking and software process improvement (10 points)

Suppose the current bug tracking system of your company has only three fields:

- Bug description (free text)
- Bug fixing description (free text)
- Name of the person who fills in the information

Your company wants to achieve the following goals through improving data collection and data analysis of the bug tracking system:

- Reduce the number of severe bugs
- Improve bug-fixing efficiency

Your task is to use the GQM (Goal-Question-Metrics) approach (and possibly combined with the IBM Orthogonal Defect Classification) taught in this course to propose data fields to be added to the bug tracking system to collect the data for achieving the goals. You should also briefly explain how you are going to analyze the collected data.

Problem 6 – OSS and software Ecosystem (10 points)

Suppose you have developed a software education game to teach kids mathematics, and you have published the software as open source. Based on what you have learned from the OSS lecture, propose your strategies to attract more open source developers to help you grow the software game fast and successfully.

Your strategies should cover the following aspects. You should briefly explain what strategies you choose to use and why you choose a particular strategy.

- License type (1 point)
- Software engineering practices (5 points)
- Social process (4 points)

Introduction

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

You need to put all your answers to the following problems 1 to 4 in a .pdf file and upload the .pdf file to Inspera before the examination time expires. Your answers to problems 1 to 4 should be brief and to the point.

Problem 1 – Use case (5 points)

Suppose you will develop an online pizza order system. Based **on the use case template taught in this course, write a use case** "order pizza online."

Ordering pizza online means that:

- The user can search pizza types online
- The user can choose the type and the amount of the pizza to order
- The user can specify the delivery address and time
- The user can provide payment card information
- The system can communicate with the bank to check the correctness of the payment card information and charge the card
- The system can issue a receipt to the users
- The system can inform the delivering company of the user's order so that the delivering company can come to pick up the pizza for delivery

Problem 2 – Goal-oriented requirement engineering (5 points)

Based on this course's goal-oriented requirement engineering lecture, the task is to make a goal model of the "autonomous bus" control system using soft goal types. The goal model should have at least three layers.

(Note: You can use whatever software to draw the goal model. Handwriting drawing is not acceptable).

Problem 3 – Software testing (10 points)

Suppose you will lead the system test of an e-voting system. Your task is to explain to your top-level managers the purpose of each type of the following system test and give them at least one test case example.

- Scalability test
- Stress test
- Load and stability test
- User acceptance test
- Operational acceptance test

Problem 4 – Code review and testing (10 points)

Use code smells your group identified and refactored in the exercises as examples to explain code refactoring.

- You need to give at least four code smells your group identified through code review as examples. Code smells of different categories are preferred.
- List the code of the four code smells before and after refactoring and explain the purpose of each refactoring.

DO NOT FORGET TO ANSWER THE CLOSED ENDED QUESTIONS (20 POINTS)