Part 1

Exercise 1

p1 is the fact that you only show the presence of defects that you are testing for.
Example: The client in the book finds only 99,5% of the defects in the software tests.

p2 is that exhaustive testing is impossible. Meaning that you can never find every defect a system may contain. Instead of testing every part of the system, the tests should follow the priorities and risks of the system being developed. (closely linked to **p6**)
Example: You do not have time to test everything, so in order to so much defects as you can you have to prioritize the testing

p3 is that you should test as early as possible in the development lifecycle of the system. Finding defects early is achieved by following this principle. (This reduces the costs in resources and time overall)
Example: The amount of cost doubles for each step the defects is «brought» through, so in order to limit the cost you have to start early.

p4 is that defects that show themselves have the tendency to be clustered with other defects, this is possibly due to the programmer being tired, unconcentrated or other factors that makes it hard for the programmer to write good code in one place at one time.
Example: Often, each of the clusters contains 80% of the defects with 20% of the modules.

p5 The pesticide paradox principle describes the blindness that we get when we continue running the same tests, this relates heavily to **p1**, that if you continue to test and find the same defects you have to write new tests to find new defects. It also relates to **p7**.
Example: If you have a system which you run tests on fix the defects in the system, and then write some more on the system and then use the same test case to find defects you would not find any defects but if you write a new test you would most likely find a new defects.

p6 Testing is context dependent, meaning that you should customize what and how you test different systems.
Example: You wouldn't reuse tests from a financial system on a plane-system.
p7 If the users needs and expectations are not satisfied it does not help to find and fix defects.
Example: If you want to buy a tv-box, but can not watch you're the channels you want to watch you most likely would not buy that television-box even though it's free from defects.

Exercise 2

A. The V-model is based upon **p3**, that we should test as early as possible. We begin testing a level as soon as the development at that level is started and we continue to test until we reach the end of each level. At each stage in the life cycle we use different test levels such as component testing, integration testing, system testing and acceptance testing in parallel we start the related tests. The test levels can be combined or reorganized depending on the nature of the project or the system architecture. This gives us an advantage over the waterfall model.

Benefits: The main benefit of the V-model is that compared to other models it's easy to implement due to the easy-to-understand structure of the model.

The V-model is more effective than the waterfall-model, this is related to the cost and time of going back through the levels to fix things that could've been found early if we only used the V-model.
The time used in development is drastically reduced using the V-model compared to the traditional waterfall model. The V-model makes it easier to validate that the product we're making is right (focus of **p7**) and verifying that we're making the product correctly.

B. Yes it is possible, but each level and the time for each sprint would be much shorter, and as the project progresses we have to do more and more regression tests to see if any new addition doesn't ruin old functionality. Agile emphasizes testing and quality throughout each iteration, not just at the end. The V-model has grooming User story at start of each iteration and has a Test driven development.

An agile best practice is the elaboration of user stories and acceptance criteria into tests. The acceptance criteria is derived in the V-model as it identifies test conditions and test oracles before coding. Test cases in the V-model is derived before coding. Exit criteria is derived in the V-model, this should not exist in the agile development.

The agile development conclude the increment based upon process activities, work products and time. The biggest advantage of the agile model is its capability to adapt to the varying requirement the development project.

Concluding that agile processes are heavily influenced by the V-model, but taking shortcuts on some aspects that are not beneficial to the time and project.

C. The V-model exercises **p2**, because testing is only limited by the resources and time and not by where in the development process we are, and we get more tests that are better at targeting specific issues based on the context of the current test level, corresponding to **p6**. It also exercises **p3** as explained in A, and the early acceptance testing makes it easier to validate the system following **p7**.

D. A test-level is defined as to **what** we're testing, clearly setting the boundaries between the different levels. In the V-model a test-level should correspond with the stages of the development-process.

Acceptance-testing: Testing and validating that were creating the right system, and that the end-user and stakeholders are happy with the system we have created and are creating.

System-testing: Testing the system as a whole to see that the whole system does as intended by the development-team(s). Also looking at the integration of the whole system, subsystems and foreign systems defined by the scope of either a development project or product.

Integration-testing: Integration testing is interested in the specific interactions between interfaces of components and even whole systems.

Unit-testing: Unit testing is tested in isolation from the rest of the system. Testing the specific implementation of each individual component in the system. (i.e classes, functions, modules) separately from one another.

A test-type is **what** we're testing **for**.

New functionality (functional testing): Testing that the system does do the functional requirements specified in the specification correctly.

Old functionality (testing change): Regression-testing and confirmation-testing tests the consequences that might occur when we're performing changes on the system.

What it does (non-functional testing): Testing that the system adheres to the non-functional requirements that stakeholders and developers has specified, like efficiency and maintainability of the system and code.

Structural tests: Testing the thoroughness of how we're testing the system and components.

** **Note**: There is some overlap between the two, but when talking about integration in system-testing we're testing all the interactions as a whole, and in integration testing we're talking about each specific interaction between systems and/or components

Exercise 3

A. Static techniques tests the code (documents) without executing it, reading (presenting) the documents and finding mistakes.

With a dynamic testing technique the software is executed using a set of values, the result of the execution is then examined and compared to the expected result. The main difference between the two, lying in the execution part and the defects that they find and that they're designed to find.

B. **Planning:** This involves planning everything that should be tested, who should test what (selecting the people) and the roles of the participants, entry-criteria and exit-criteria and also how the review meeting and follow-up should be handled.

Kick-off: This involves distributing the documents (programs) that should be reviewed to the right people. It also involves letting the people know their different responsibilities and presenting them the material and process. Ensuring that the participants are on the same wavelength and have the same understanding of the rest of the process.

Preparation: This is a time consuming process in which each participant reviews the material in the way they were instructed, documenting the defects, comments and questions that should arise.

Review-meeting: This is when each participant meets to discuss the issues and questions arised when they prepared. Documenting and discussing each question, comment and defect, as well as finding potential ways to fix the issues that they found, and maybe also issues regarding the review. Documenting and discussing is divided into two phases of the review-meeting.

Rework: Involves the author(s) fixing the defects and issues found and recording the updated status of those defects.

Follow-up: In the follow up we should check if the defects found has been corrected or at least been looked at, documenting the metrics of test and defect-fixes. In the last part we should check the exit-criteria to see if the criteria has been met indicating whether we could end our test or if we have more work to do.

C. **Walkthrough:** The meeting is led by the author(s) and the content of the document is explained step by step by, to reach consensus on changes or to gather information. The benefit of having a walkthrough is that the author gets a more objective view and constructive criticism from cross-functional group of people. This is not a very formal type of review and can be done in a short amount of time.

I**nspection:** Is a type of peer-review where the participants examines the documents visually making sure that the documents adhere to standards and rules also making sure that the documents do not conflict with higher-level documents. This may happen before the meeting or during the meeting. This is a very formal type of review since non-adherence to the requirements carry a large amount of impact.

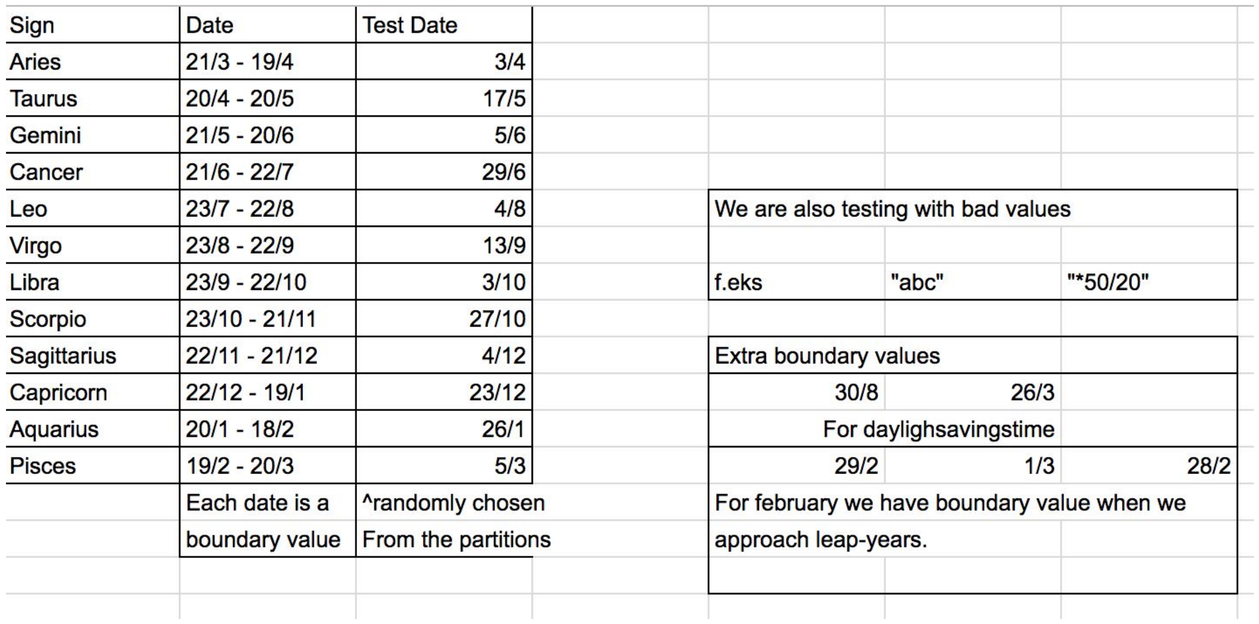
Inspections are more focused on finding compliance in the documents, making sure that they do not break any laws or important formal requirements. Walkthroughs are more focused on if the documents are well made.The purpose and goal of the inspection is to fulfill the requirements and adhere to laws and improve the quality of the system as whole. Walkthroughs are more focused on the individual documents “superficial” quality. Concluding that formality and the elements being inspected (the goals) are very different.

Exercise 4

1. «Black-box»-testing tests the output based on an input and an expected value. It’s called that because the interesting part of what we’re testing isn’t the structure or how it is implemented, but if it does what it’s supposed to do based on the specification.
   Visually a black box is impossible to see through so imagine that the object inside is a component or function where what we’re interested in is outside the box.
2. «White-box»-testing also called «glass-box»-testing, is the opposite, we’re interested in the structure of the implementation and how much of the code (document) is covered by tests.
   Visually a white-box(glass-box) is see-through so we can see what’s inside the box, and that is the interesting part for us when we’re white-box-testing
3. Experienced-based testing can be very effective, but it depends solely on the person(s) who is testing (reviewing) the documents. Human error is very much a drawback of experience-based techniques. The benefits is that it might find defects that we don’t have techniques to test for. It can also be much quicker instead of writing automated or new tests.
4. We’ll refer back to testing-principle number 6 «testing is context dependent» and it should be treated as such. By saying that it should be clear that having an opinion before knowing the context would be redundant and superficial.

Part 2

Exercise 5



Sign	Date	Test Date
Aries	21/3 - 19/4	3/4
Taurus	20/4 - 20/5	17/5
Gemini	21/5 - 20/6	5/6
Cancer	21/6 - 22/7	29/6
Leo	23/7 - 22/8	4/8
Virgo	23/8 - 22/9	13/9
Libra	23/9 - 22/10	3/10
Scorpio	23/10 - 21/11	27/10
Sagittarius	22/11 - 21/12	4/12
Capricorn	22/12 - 19/1	23/12
Aquarius	20/1 - 18/2	26/1
Pisces	19/2 - 20/3	5/3
	Each date is a boundary value	^randomly chosen From the partitions

We are also testing with bad values		
f.eks	"abc"	"*50/20"

Extra boundary values		
30/8	26/3	
For daylighsavingstime		
29/2	1/3	28/2
For february we have boundary value when we approach leap-years.		

A. Test Date is the date that we have chosen from each partition.

B. Despite the general boundary values we have some special boundary values that relates to the calendar rather than the signs.

C. We found some defects. Assume that highest priority is 3 and lowest priority is 1.

Priority	Defect
Low	If a user writes a string instead of a proper number, there is no message describing what the user is doing wrong.
Medium	If a user does not write anything or only writes in one field, there is not a describing message of what the user is doing wrong.
High	12/21 is an invalid date as a formal input
Low	You can insert mathematical operators at the start or end of the input for both the day and month field.

D.
Identifier: 1
Severity: Medium
Impact: Medium
Date: 21.03.2017
Tester: Peder
Program: YourHoroscope.jar

Summary: If you only enter a number in one field and submit, you don't get a message of what went wrong
Input:

- Month:
  Date: 3 (<a number>)

- Month:3 (<a number>)
  Date:

Detailed description:
Input: If you only enter a month or only a day.
Output: You get a bad description of what is happening not describing the problem. But you will not get an answer and that is correct.
Expected: A good description that you need to enter something into both fields.
Repeatment: I repeated the test multiple times with several inputs and got the same result for every good and bad input in each field separately.
Contributions: Robin performed the inputs that I gave him and contributed to the test.

Identifier: 2
Severity: Low
Impact: Medium
Date: 21.03.2017
Tester: Robin
Program: YourHoroscope.jar
Summary: The program accepts characters, mathematical operators and expressions in the date and time field.
Input:

- Month: abc
  Date: def

- Month: *3
  Date: *6

- Month: 3/10
  Date: 10/3

Detailed description: If given a character, mathematical operator or expression in the date and day field and followed by pressing the "See your horoscope" button, you will not get an error message, but a generic message description.

Input: Given alphabetic character in date and time field
Output: Message description explaining that this should not be a valid input.

Input: Given a mathematical operator followed by a number in the date and time field

Output: Same message description as above

Input: Given a mathematical expression in the number and time field
Output: Same message description as above

Expected result: Error message describing that the input types is not valid.

Identifier: 3
Severity: Medium
Impact: High
Date: 21.03.2017
Tester: Cindy
Program: YourHoroscope.jar
Summary: 12/21 is an invalid birthdate
Test Procedure used to discover the incident.

- Open the program.
- Enter 12 in the field «In what month are you born»
- Enter 21 in the field «What day of the month are you born?»
- Push «See your horoscope»

Detailed description:
Input:

- Enter 12 in the field «In what month are you born»
- Enter 21 in the field «What day of the month are you born?»

Output:

- «You have choosen an unvalid date for your birthday! Try again!»

Anomalies:

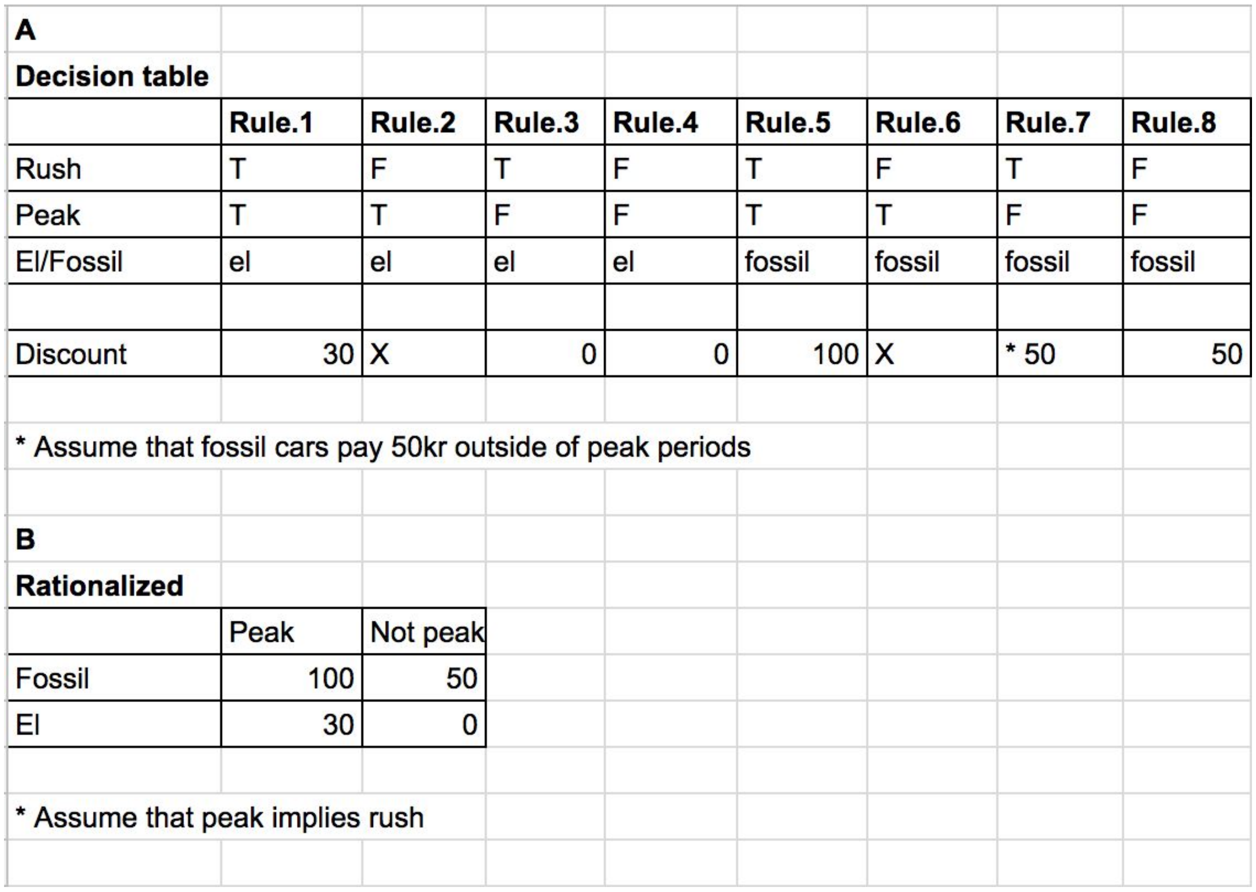
- The user born on 21/12 do not get it's horoscope, even though this is a legal date.

Procedure step:

- Enter 12 in the field «In what month are you born»
- Enter 21 in the field «What day of the month are you born?»
- Push «See your horoscope»

Expected results: Get «Your sign is sagitarius Here is your horoscope: ..» as an output

Exercise 6



A

Decision table

	Rule.1	**Rule.2**	**Rule.3**	**Rule.4**	**Rule.5**	**Rule.6**	**Rule.7**	**Rule.8**
Rush	T	F	T	F	T	F	T	F
Peak	T	T	F	F	T	T	F	F
El/Fossil	el	el	el	el	fossil	fossil	fossil	fossil
Discount	30	X	0	0	100	X	* 50	50

* Assume that fossil cars pay 50kr outside of peak periods

B

Rationalized

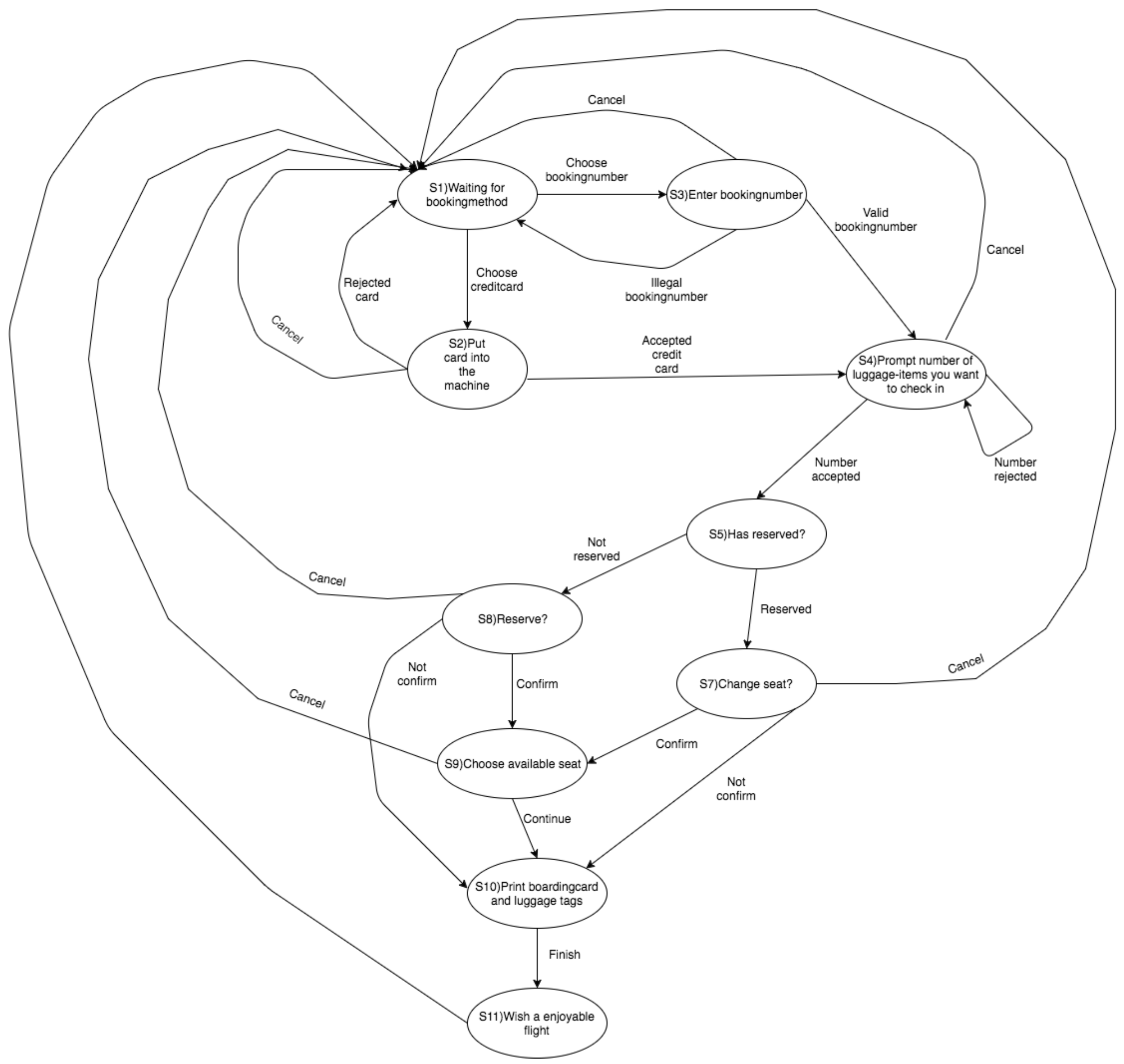
	Peak	Not peak
Fossil	100	50
El	30	0

* Assume that peak implies rush

Part 3

Exercise 7

A.



B. See attachment7b.pdf for state transition table

C.

I: You measure how many statements that have been tried compared to how many statements that exist.

II: To measure transition coverage you measure how many transitions from each statement you have tried compared to how many transitions that exist.

D.

Beginning from the statement: Waiting for booking

- Choose bookingnumber
- Valid bookingnumber
- Number accepted

The 5 transition-groups to be tested after the first three:

1.

- Reserved
- Confirm
- Continue
- Finish

2.

- Not reserved
- Not confirm
- Finish

3.

- Choose bookingnumber
- Valid bookingnumber
- Number accepted

4.

- Reserved
- Not confirm
- Finish

5.

- Choose bookingnumber
- Illegal bookingnumber

Afterwards:

- Choose creditcard
- Rejected card
- Choose creditcard
- Accepted creditcard
- Number rejected
- Number accepted
- Not reserved
- Confirm
- Continue
- Finish

** Note: Trying cancel from each state has to be done as well

The statement coverage is 100%.

If you achieve a 100% transition coverage you will automatically visit each state and thereby have 100% statement coverage.

Exercise 8

A. The focus should be on the main interaction between the user and the system. One scenario would be in the interaction when the user receives the luggage tags . Also when the user has/has not reserved seats. In addition, when the user wishes to use booking number and the user has/has not reserved seats.

	Step	Description
Main success scenario A:Actor S: System	1	S: Prompts which booking method want to use
	2	A: Choose booking method
	3	A: Enters invalid booking number
	4	A: User cancel
Extensions	4a)	User enters invalid booking number. User tries again

	Step	Description
Main success scenario A:Actor S: System	1	S: Prompts which booking method want to use
	2	A: Choose credit card method
	3	A: Put card into machine
	4	S: Do not accept credit card
Extensions	3a)	A: Cancel. System returns to choose booking method.

	Step	Description
Main success scenario A: Actor S: System	1	S: Prompts which booking method want to use
	2	A: Choose credit card method

	3	S: Accepts card
	4	S: Prompt number of luggage items to check in
	5	A: User enters luggage items to check in and have not reserved seat before.
	6	S: Ask if user want to reserve seat
	7	A: User do not confirm
	8	S: Prints out luggage tags
	9	S: Wish enjoyable flight
Extension	3a)	S: Credit card not accepted S: Returns to point 1.
	3b)	A: Push cancel. S: Returns to point 1.
	4a)	S: Reserved seat before U: Do not confirm.
	4b)	S: Reserved seat before U: Do confirm S: Display available seats U: Choose available seats
	4c)	S: Have not reserved seat before U: Do confirm S: Display available seats U: Choose available seats

	Step	Description
Main success scenario	1	S: Prompts which booking method want to use
	2	A: Choose booking method method
	3	S: Accepts booking number

	4	S: Prompt number of luggage items to check in
	5	A: User enters luggage items to check in and have not reserved seat before.
	6	S: Ask if user want to reserve seat
	7	A: User do not confirm
	8	S: Prints out luggage tags
	9	S: Wish enjoyable flight
Extension	3a)	S: Credit card not accepted S: Returns to point 1.
	3b)	A: Push cancel. S: Returns to point 1.
	4a)	S: Reserved seat before U: Do not confirm.
	4b)	S: Reserved seat before U: Do confirm S: Display available seats U: Choose available seats
	4c)	S: Have not reserved seat before U: Do confirm S: Display available seats U: Choose available seats