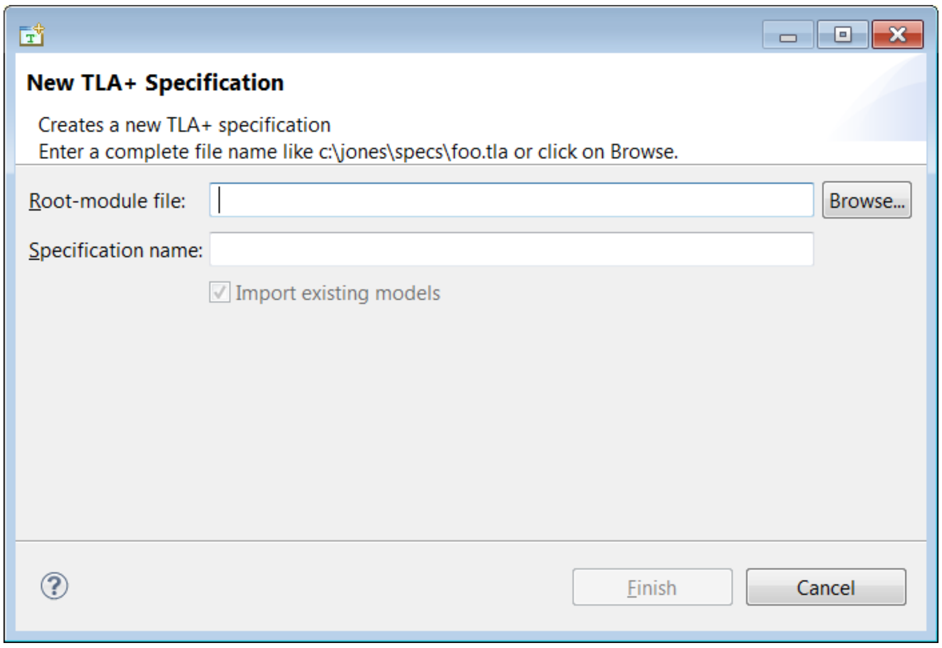
BCS2213 – Formal methods

Teaching assignment 2. Specification of a simple system using TLA+ Toolbox.

1. Run TLA+ Toolbox and click on button (left panel) to open **Spec Explorer** window.

(**Note.** TLA+ Toolbox can be downloaded from https://tla.msr-inria.inria.fr/tlatoolbox/products/)

2. Inside **Spec Explorer** window click by the right mouse button and choose from popup menu "New Specification".

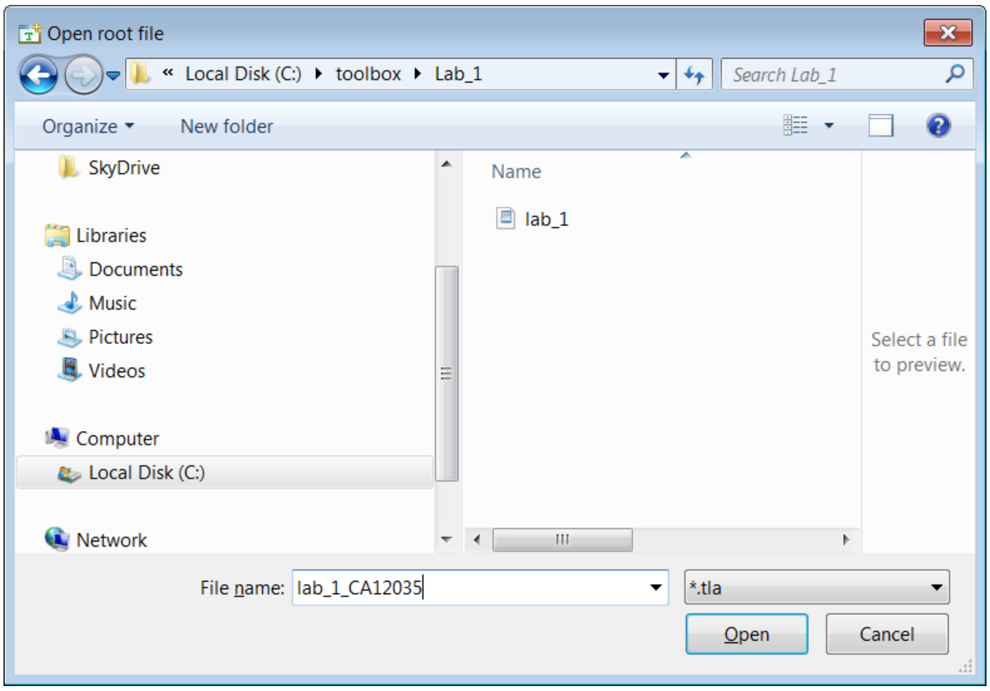

New TLA+ Specification

Creates a new TLA+ specification
Enter a complete file name like c:\jones\specs\foo.tla or click on Browse.

Root-module file:
Specification name:
Import existing models

3. Click **Browse…** and in the Formal Methods folder create your personal folder with your student Id as a name.

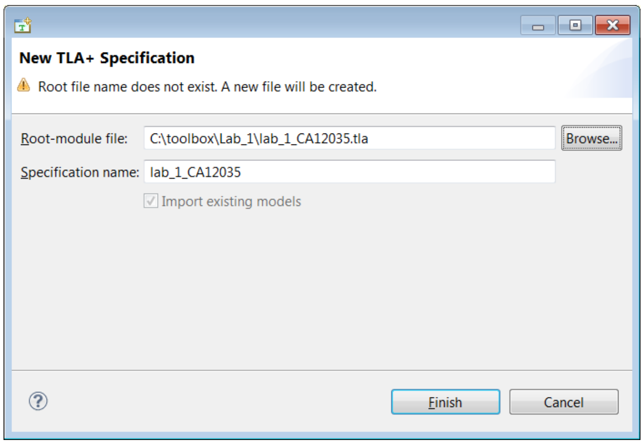
4. Inside this folder create the folder with the name "lab_2" and next specify the file name "lab_2_<your_ID>", an example is shown in the figure below (the extension .tla will be automatically added to the file).

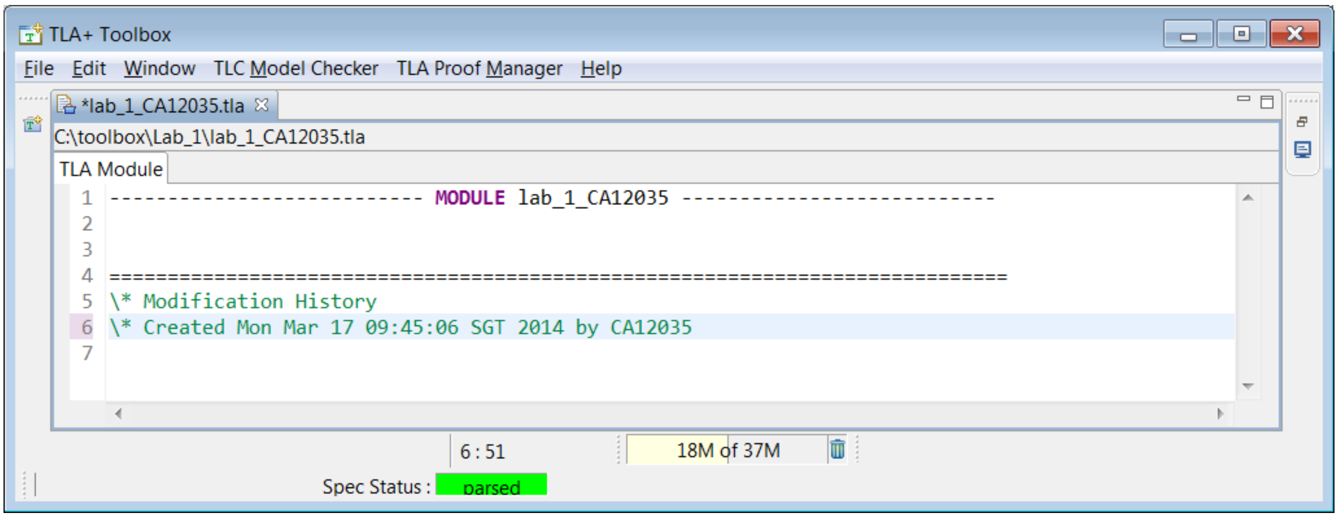

Open root file

« Local Disk (C:) ▸ toolbox ▸ Lab_1

File name: lab_1_CA12035
*.tla

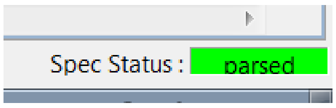
5. Click **Finish**. As a result, you will see newly created TLA module lab_2_<your_ID>.


New TLA+ Specification
Root file name does not exist. A new file will be created.
Root-module file: C:\toolbox\Lab_1\lab_1_CA12035.tla
Browse...
Specification name: lab_1_CA12035
Import existing models
Finish
Cancel

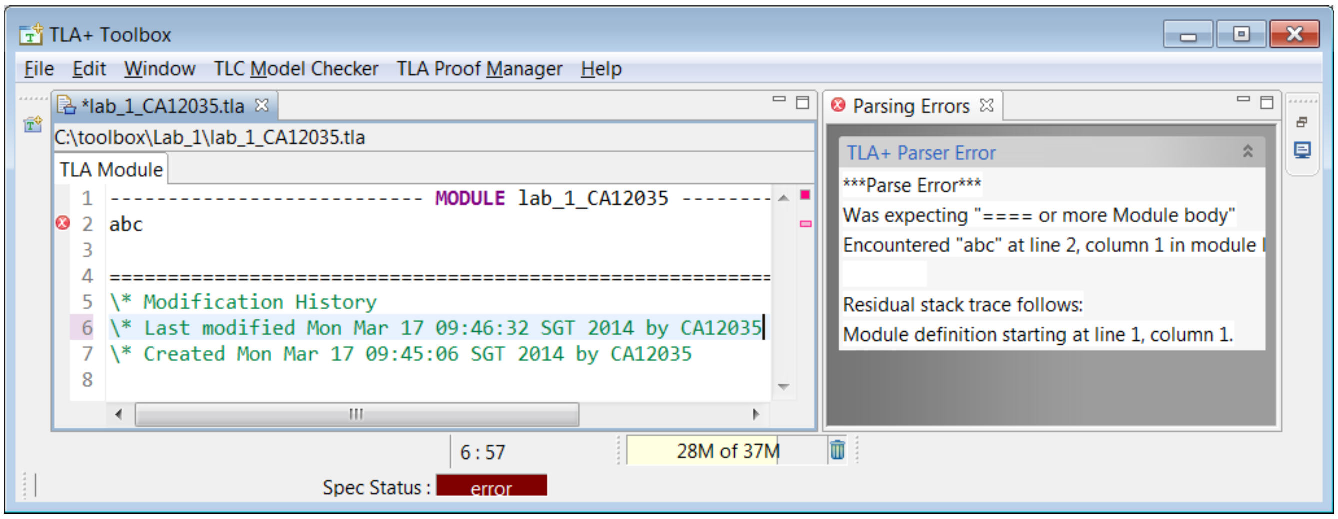

TLA+ Toolbox
File Edit Window TLC Model Checker TLA Proof Manager Help
*lab_1_CA12035.tla
C:\toolbox\Lab_1\lab_1_CA12035.tla
TLA Module
6 : 51
18M of 37M
Spec Status : parsed

```
---------------------------- MODULE lab_1_CA12035 ----------------------------



=============================================================================
\* Modification History
\* Created Mon Mar 17 09:45:06 SGT 2014 by CA12035

```

6. Have a look on the status of your module in its right bottom corner. It is parsed and if module contains no errors, the status is highlighted by green colour.



7. Write “abc” in the line 2 and reparse module (use File -> Parse Module or press Ctrl+R). You will see the error message



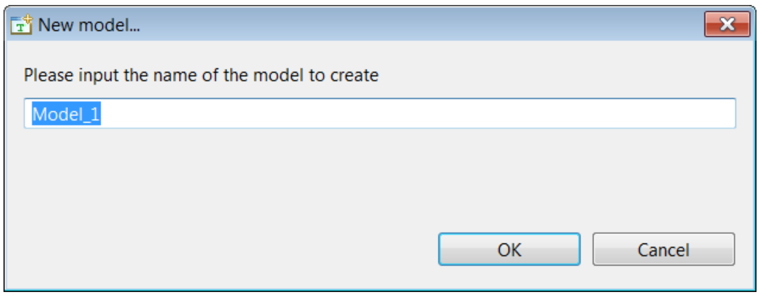
```
---------------------------- MODULE lab_1_CA12035 --------
abc

===========================================================
\* Modification History
\* Last modified Mon Mar 17 09:46:32 SGT 2014 by CA12035
\* Created Mon Mar 17 09:45:06 SGT 2014 by CA12035

```

```
***Parse Error***
Was expecting "==== or more Module body"
Encountered "abc" at line 2, column 1 in module l

Residual stack trace follows:
Module definition starting at line 1, column 1.
```

8. Delete the erroneous “abc” and reparse the module to have the “green” status.

9. Create the new model of specification in TLC Model Checker (use the main menu item TLC Model Checker -> New model…).

By default the model name is “Model_1”



Please input the name of the model to create

Model_1

The Toolbox uses the TLC to check the current specification. More precisely, TLC checks a *model* of the specification.

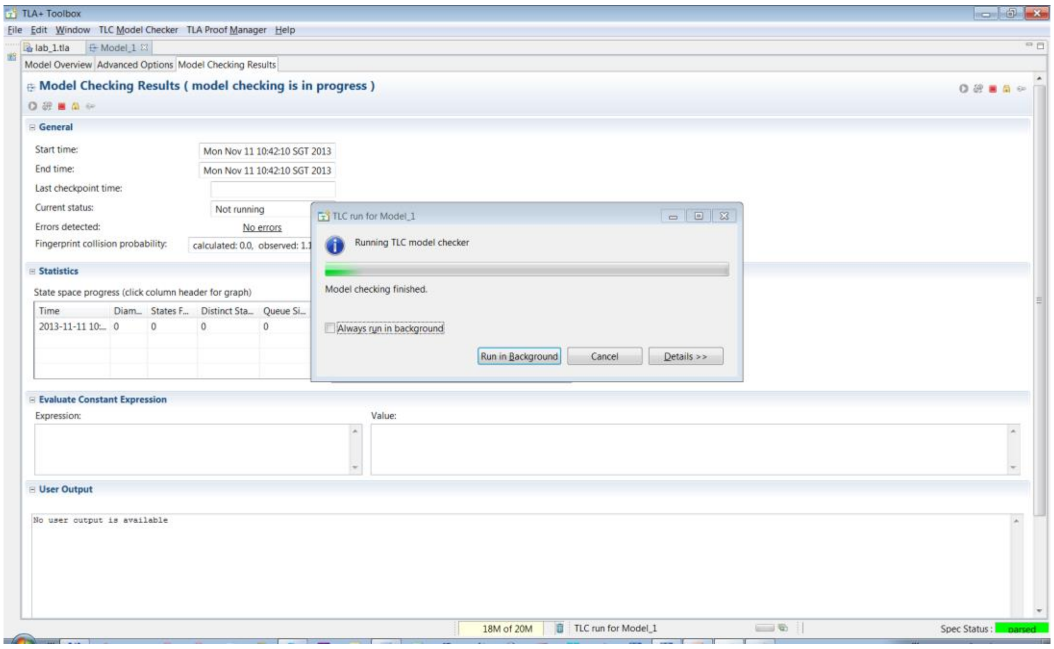
Model editor has three pages:

- Model Overview Page
- Advanced Options Page
- Model Checking Results Page

You can edit these pages in the usual way. Sections can be opened or closed by clicking the + or - . When entering text in fields, you can use your system's standard editing commands.

The Toolbox does some validation of the model as you edit it. If that validation finds no errors, clicking on *validate* ( ) or *run* ( ) checks for other errors. The Toolbox reports any errors in the model that it finds by placing error balloons like this near the part of the model containing the error. Moving the mouse cursor on top of an error balloon will raise a message explaining the error. The Toolbox also puts an *Errors Detected* field at the top of the page. Moving the mouse cursor on top of it raises all the error messages for the page.

10. Press green arrow button to run TLC (Temporal Model Checker) on the model.



Model Checking Results (model checking is in progress)

General

Start time:	Mon Nov 11 10:42:10 SGT 2013
End time:	Mon Nov 11 10:42:10 SGT 2013
Last checkpoint time:	
Current status:	Not running
Errors detected:	No errors
Fingerprint collision probability:	calculated: 0.0, observed: 1.1

Statistics

State space progress (click column header for graph)

Time	Diam...	States F...	Distinct Sta...	Queue Si...
2013-11-11 10:...	0	0	0	0

Evaluate Constant Expression

Expression:

User Output

```
No user output is available
```

Running TLC model checker

Model checking finished.

Always run in background

18M of 20M

TLC run for Model_1

Spec Status : parsed

11. Learn the results of evaluation. For your *empty* model TLC have found *zero* states and *zero* distinct states. Note,

States Found - The total number of states TLC has examined so far.
Distinct States - The number of distinct states among the states found.

12. Specify the HourClock system, following the next TLA definitions in TLATEX format

$$\text{MODULE } \mathit{HourClock}$$

$$\text{EXTENDS } \mathit{Naturals}$$

$$\text{VARIABLE } \mathit{hr}$$

$$\mathit{HCini} \triangleq \mathit{hr} \in (1 \,..\, 12)$$

$$\mathit{HCnxt} \triangleq \mathit{hr}' = \text{IF } \mathit{hr} \neq 12 \text{ THEN } \mathit{hr} + 1 \text{ ELSE } 1$$

$$\mathit{HC} \triangleq \mathit{HCini} \land \Box[\mathit{HCnxt}]_{\mathit{hr}}$$

$$\text{THEOREM } \mathit{HC} \Rightarrow \Box \mathit{HCini}$$

13. Please note, in TLA+ Toolbox you need use the ASCII equivalent of TLA notation (see next figure).

```
---------------------- MODULE HourClock ----------------------
EXTENDS Naturals
VARIABLE hr
HCini  ==  hr \in (1 .. 12)
HCnxt  ==  hr' = IF hr # 12 THEN hr + 1 ELSE 1
HC  ==  HCini /\ [][HCnxt]_hr
--------------------------------------------------------------
THEOREM  HC => []HCini
==============================================================
```

Explanations:

- the variable **hr** represent the clock's display from 1 to 12
- the initial predicate is formula **HCini**, specifying the possible value of hr
- the next-state relation **HCnxt** is a formula expressing the relation between the values of **hr** in the previous state and the next state of a system
- the next-state relation is that **hr'** equals **hr+1** except **if hr** equals **12**, in which case **hr'** should equal **1**.
- HCnxt is an ordinary mathematical formula, except that it contains primed and unprimed variables. Such a formula is called an **action**.
- an action formula (HCnxt) can be true or false of a step.
- the specification of the hour clock is the definition of temporal formula **HC**, including the definitions of the both formulas HCnxt and HCini

$$\mathit{HC} \triangleq \mathit{HCini} \land \Box[\mathit{HCnxt}]_{\mathit{hr}}$$

- to express HC we use the TLA operator □ (pronounced box). The temporal formula □F asserts that formula F is always true.
- so **hr** should be an integer from 1 through 12 in every state of any behaviour, satisfying the clock's specification, i.e. the formula HC
- a temporal formula satisfied by every behavior is called a theorem, so **HC => □ HCini** is a theorem. Or, in TLATEX notation

$$\text{THEOREM } HC \Rightarrow \Box HCini$$

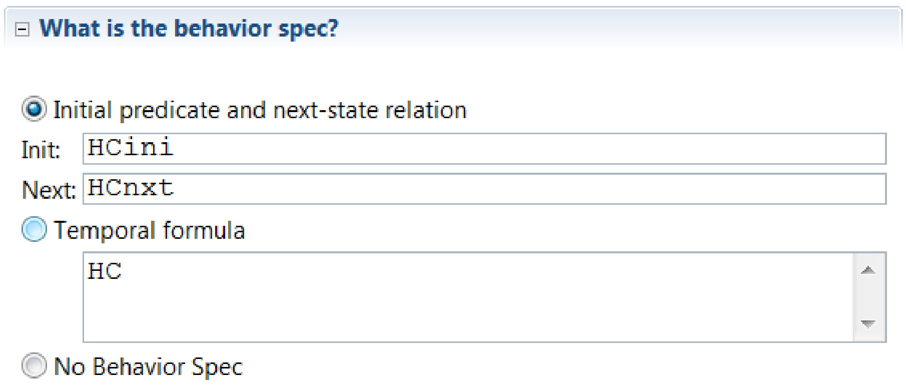
14. Specify in the Model overview page *what* to check.

There are two ways to write the behaviour spec:
Initial predicate and Next-state relation
A pair of formulas (Init and Next) that specify the initial state and the next-state relation, respectively

OR

Temporal formula
A single temporal formula of the form `Init /\ [][Next]`$_{\texttt{vars}}$, where `Init` is the initial predicate, `Next` is the next-state relation, `vars` is the tuple of state variables, e.g. `HCini /\ [][HCnxt]_hr`

Specify the behaviour by initial predicate (HCini) and the next state relation (HCNext).


What is the behavior spec?

[● Initial predicate and next-state relation]
Init: HCini
Next: HCnxt
○ Temporal formula
HC
○ No Behavior Spec

15. Run TLC on the HourClock specification and see results.

How much states and distinct states TLC have found for your model? Why?

Write explanations inside your Lab_2 module.

16. Modify HCini and HCnxt

```
HCini  ==  hr \in (1 .. 11)
HCnxt  ==  hr' = IF hr < 11 THEN hr + 1 ELSE 1
```

and check the results with TLC. How much states and distinct states TLC have found now?

What is your explanation? Write your explanations inside your Lab_2 module

17. Write specification in the alternative form (you can do it in the same file or create new specification. If you use the same file, just comment your previous specifications, but not delete it).

$$HCnxt2 \triangleq hr' = (hr \ \% \ 12) + 1$$

$$HC2 \triangleq HCini \land \Box[HCnxt2]_{hr}$$

Where % is the modulus operator.

18. Check the HC2 specification. Is amount of states and distinct states changed?

19. Click on the **+ What to check** in the model overview. There are three kinds of properties of the behaviour that TLC can check:

Deadlock. A *deadlock* is occur in a state for which the next-state relation allows no successor states. Termination is deadlock that is not considered an error. If you want the behavior to allow termination, then you should uncheck the deadlock option.

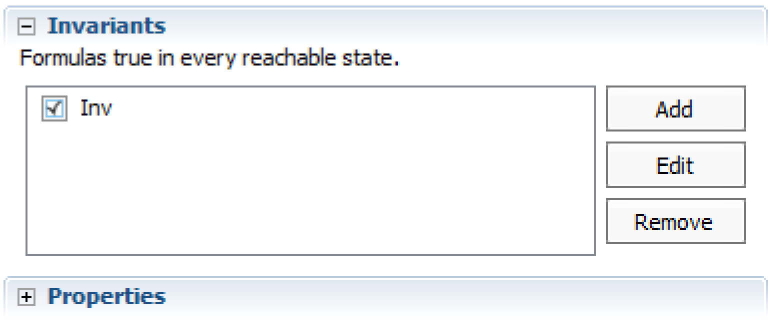
Invariants. An invariant is a state predicate that is true of all reachable states--that is, states that can occur in a behavior allowed by the specification. You can include a list of invariants. The checking of each invariant can be enabled or disabled by checking or unchecking its box.

Properties. TLC can check if the behavior satisfies (implies) a temporal property, which is expressed as a temporal-logic formula. You can specify a list of such properties, each with a check-box for enabling or disabling its checking.

20. Add in the specification a simple Invariant, telling that the value of hr is more than (equal to) 1 hour and less than (equal to) 12 hours

```
Inv == hr >= 1 /\  hr <= 12
```

Add this invariant into model overview page


Invariants
Formulas true in every reachable state.
✓ Inv

Check the model. **Note**. You may use invariant predicate as a part of your specification

```
HC  ==  Inv /\ HCini /\ [][HCnxt]_hr
```

Or theorem

```
THEOREM  HC => []HCini  /\ Inv
```

21. Try to violate invariant by specifying not correct border values for hours. Write your explanations and comment the invariant.

22. Model deadlock behaviour of your system. For it, define the next state relation as

```
HCnxt  ==    /\ hr < 12

             /\ hr' = hr+1
```

Write your explanations and comment (not delete) this leading to deadlock next state relation.

Restore correct next state relation.

23. Modify your specification by adding variable `min` (minutes). Please note, you need modify both initial and next state predicates. Also, your next state predicate will already depend on two variables – `hr` and `min` (the tuple `<<hr, min>>`). Check with TLC, how much states you have now. Explain this property inside your lab.

24. Upload results of your work (i.e. the file lab_2_<your_ID>.tla) into Kalam. If you have several files, please zip it into lab_2_<your_ID>.zip. It will be evaluated in 2.5% of your general marks.