CSC 148: Introduction to Computer Science

Writing Skills and Expectations

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Continuing the A1 Discussion

- On Friday, we introduced the *code* in A1.

- Today, we're briefly going to discuss the *writing expectations* in A1.

Overview

- Structure and organization
- Writing Mechanics
- Understanding Audience Expectations

Structure/Organization

- Logical structure
- Effective use of transition expressions

Example

- What is wrong with this example?

We implemented running a python program in method called run_program() of the Computer class, which returns an exit code based on a program specified as the argument. To open a course website in a browser, use a Computer object's browser() method, but we considered watching a cat video except we created a separate method for that.

Example

- A better way to write this (not the only way, emphasis added):

The class Computer **provides functionality for opening the course website in a browser, running a python program, and watching a cat video.** Each of these computer actions has a method which is tasked with one and only one purpose. **First,** the browse() method takes a URL argument and returns a code indicating success or failure in opening the page. **Second**, the run_program() method …

Writing Mechanics

- Sentences are complete.
- There is no excessive use of bullet points.
  - When used it is clear how their contents relate to the sentences/paragraphs around them.
- Pronoun antecedents are clear (especially referring to when “it”, “they”, etc., are used).
- Sentences are focused on a single idea and not too long.
- Language used is descriptive and precise, rather than generic.

Example 1

- What's the issue here? Hint: look at the context of the list of bulletpoints.

As part of implementing Task 1, the difficult parts were determining how the pieces of the given starter code connect with each other. For instance, the most difficult part was testing the balance of a customer's purchase, due to various bugs in our code.

- Class PriceFinder:
  - Method muffin(): returns the price of a muffin at the Store provided as an argument.
  - Method milk(): returns the price of milk at the Store provided as an argument
- Class Person:
  - Method make_purchase(): purchases the Items provided as arguments and returns a balance

The tests we created are included in the test_functionality.py file and use the pytest format we learned in class.

Example 1

- What's the issue here? Hint: look at the context of the list of bulletpoints.

As part of implementing Task 1, the difficult parts were determining how the pieces of the given starter code connect with each other. For instance, the most difficult part was testing the balance of a customer's purchase, due to various bugs in our code.

- Class PriceFinder:
  - Method muffin(): returns the price of a muffin at the Store provided as an argument.
  - Method milk(): returns the price of milk at the Store provided as an argument
- Class Person:
  - Method make_purchase(): purchases the Items provided as arguments and returns a balance

The tests we created are included in the test_functionality.py file and use the pytest format we learned in class.

What's the connection between the bulletpoint list and the rest of the text?

Example 2

- What is primary problem in these sentences?

Class A has a list attribute X and class B has a dictionary attribute Y. They are provided in the starter code.

We implemented the method X after we implemented the __init__ method, which was harder to implement.

Example 2

- What is primary problem in these sentences?

Class A has a list attribute X and class B has a dictionary attribute Y.

They are provided in the starter code.

Who is "they"?

We implemented the method X after we implemented the __init__ method, **which** was harder to implement.

Which one? X or __init__?

Understanding Audience Expectations

- Do not use slang or text-speak.
- Do not use verb abbreviations and other informal phrasings.
- Gender-neutral and inclusive language should always be used.
- Where possible, the language should be impersonal rather than personal.

Example 1

- Identify the problems in these paragraphs:

We decided to, like, implement the function which we were told to in the handout in Task1, but we ran into a bug, lol. We couldn't fix it right away, so whatever, we went to office hours. The TA helped us and we fixed it :thumbsup:

The func returns None only if the param provided is an empty list. Otherwise, it returns a float value which is approx the result of dividing the two items in the list.

Example 1

- Identify the problems in these paragraphs:

We decided to, like, implement the function which we were told to in the handout in Task1, but we ran into a bug, lol. We couldn't fix it right away, so whatever, we went to office hours. The TA helped us and we fixed it :thumbsup:

The func returns None only if the param provided is an empty list. Otherwise, it returns a float value which is approx the result of dividing the two items in the list.

Example 2

- What's the problem in this example?

The implementation of this method is based on the Customer's action. So if he decides to pass action X as an argument, the method will do …, if he decides to pass action Y as an argument, the method will do nothing.

Example 2

- What's the problem in this example?

The implementation of this method is based on the Customer's action. So if he decides to pass action X as an argument, the method will do …, if he decides to pass action Y as an argument, the method will do nothing.

Use gender neutral and inclusive language.

e.g, "they" in this case.

Example 3

- What is one thing you would improve in these sentences?

We think that it's possible this method will perform well when the input is small, but will certainly perform poorly for large inputs.

We want the implementation to reflect the specifications from the docstrings.

Example 3

- What is one thing you would improve in these sentences?

We think that it's possible this method will perform well when the input is small, but will certainly perform poorly for large inputs.

> Use impersonal language if possible, e.g.
>
> This method will likely perform well when …

We want the implementation to reflect the specifications from the docstrings.

> Use impersonal language if possible, e.g.
>
> The implementation must reflect the …

CSC 148: Introduction to Computer Science

Week 4

Abstraction, abstract data types

University of Toronto Mississauga,

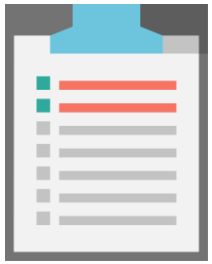
Department of Mathematical and Computational Sciences

Abstraction

- View objects as entities that can store data and operations, useful to solve problems
- Focus on semantics
  - Hides the gory details from the user
  - Freedom to design or update algorithms
  - Independent of programming language
- Examples?

Abstract Data Types (ADTs)

- In CS, we recycle our intuition about the outside world as ADTs
- We abstract data and operations, and suppress the implementation

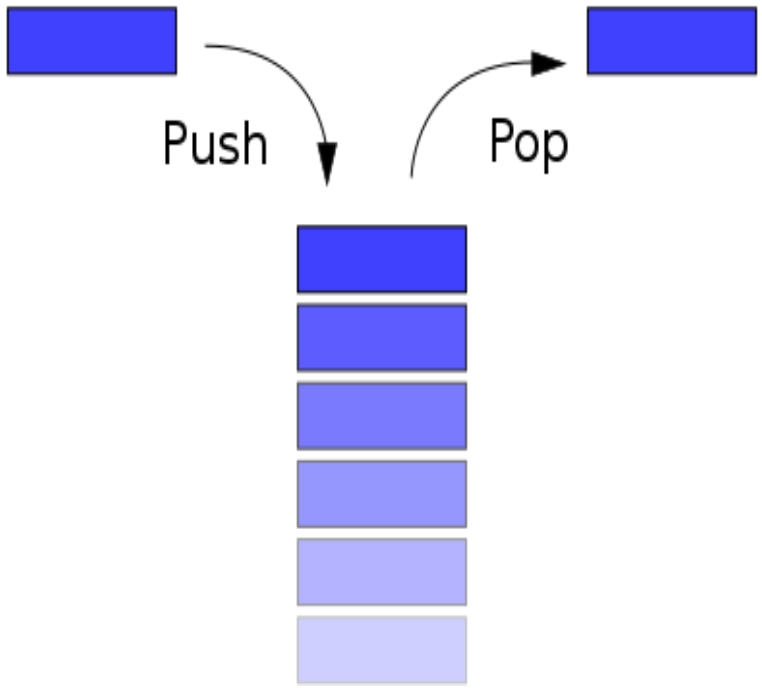


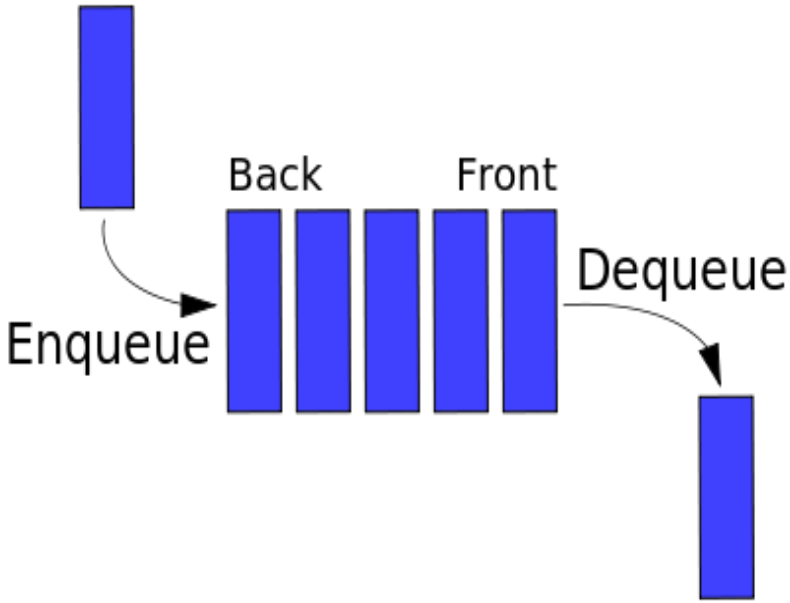
Sequences of items; can be added, removed, accessed by position, etc.

Specialized collection of items where we only have access to most recently added item

Collection of items accessed by their associated keys

Stacks and Queues





Stack ADT

- In Python, frames for function calls form a stack
- What does a stack store?
- What are the operations?

Stack application:

balanced parentheses

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Parenthesization

- In some situations, it is important that opening and closing parantheses, brackets, braces match

(a * b) + c	good
a *) b + (c	bad
(a + [b – {c*d}])	good
(a + [b – {c*d)]}	bad
(ab(cd(e)fg))(h(i(j)(k(l)))m(n))	thoughts?

- Why is this useful?
  - e.g., your IDE checks for well-formed code
  - compilers, interpreters, calculators, etc.

Define Balanced Parentheses

- A string with no parentheses is balanced
- A string that begins with a left parenthesis "(", ends with a right parenthesis ")", and in between has balanced parentheses. Same for brackets "[...]" and braces "{...}"
- The concatenation of two strings with balanced parentheses is also balanced: (...) (...)

Simplified Problem: Regular Parentheses

- Key ideas:
  - Ignore all characters except (and)
  - Keep track of when you see a (but forget about it when you've seen the matching)

- We'll give you 4 examples of expressions, one character at a time
- Use the stack on the worksheet, to determine if the Parentheses are balanced!
- Ready?

1

9

*

8

6

(1 - (5 * 8) + 6

Second expression

- Ready?

a

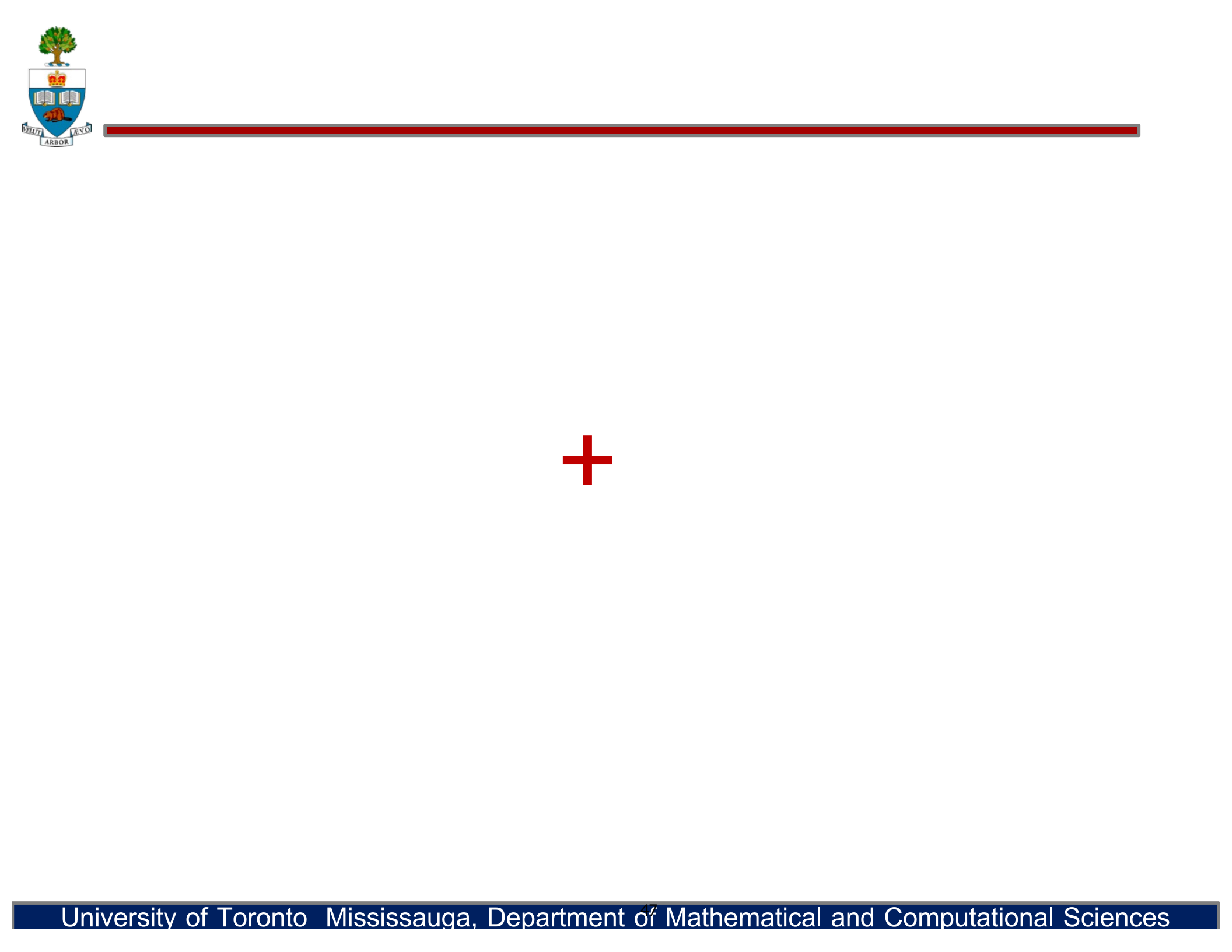

VELUT
ARBOR
ÆVO

*

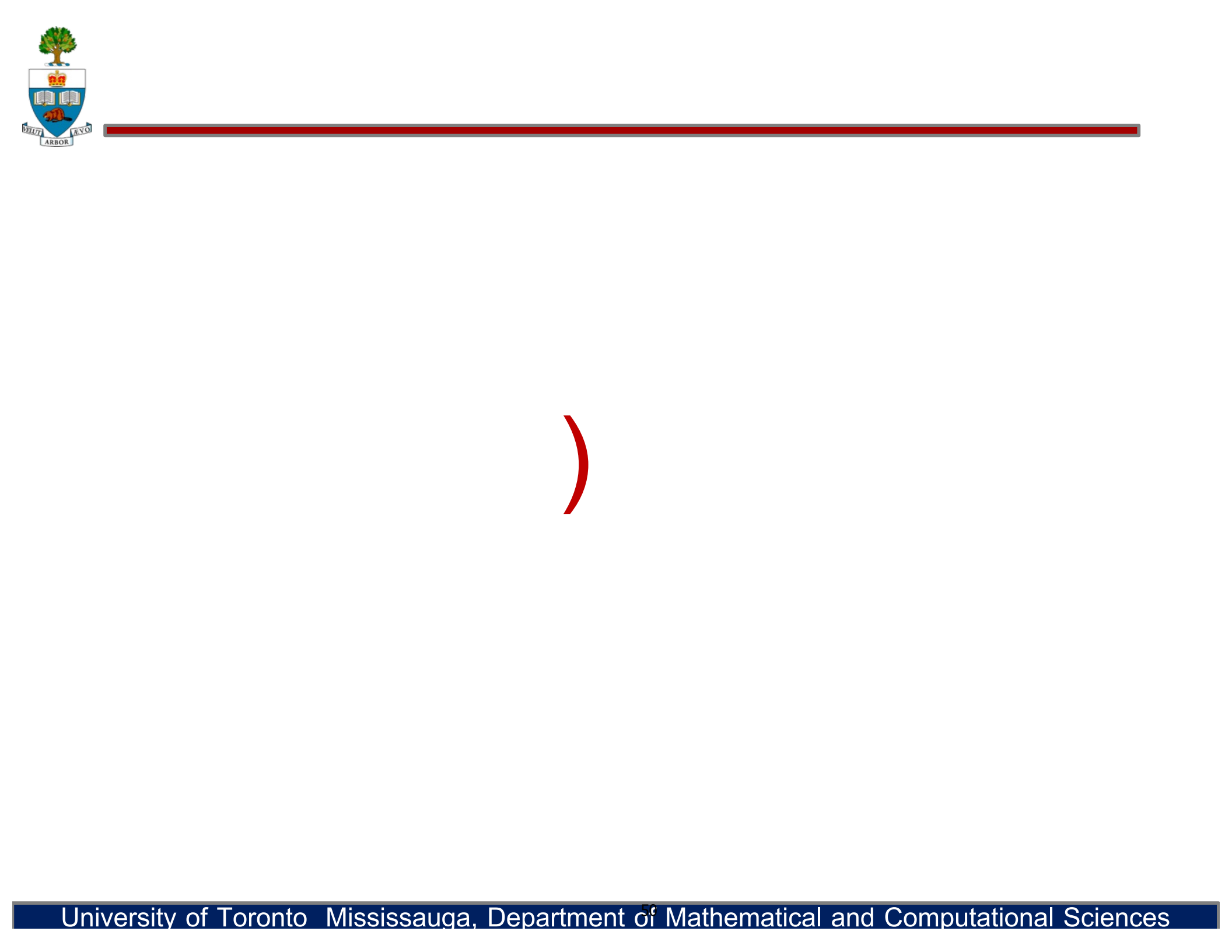
3


VELUT
ARBOR
ÆVO

)



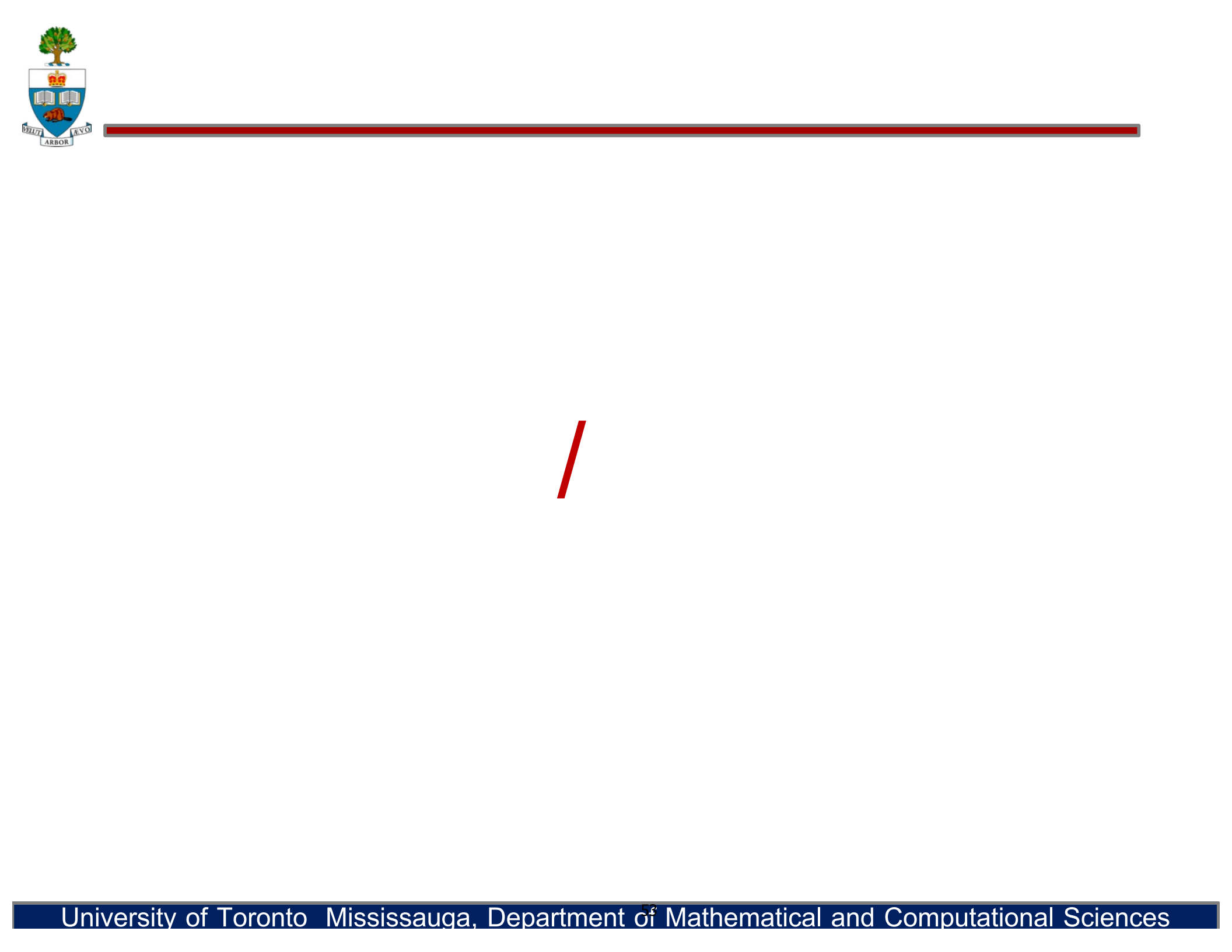
b



)


VELUT
ARBOR
ÆVO

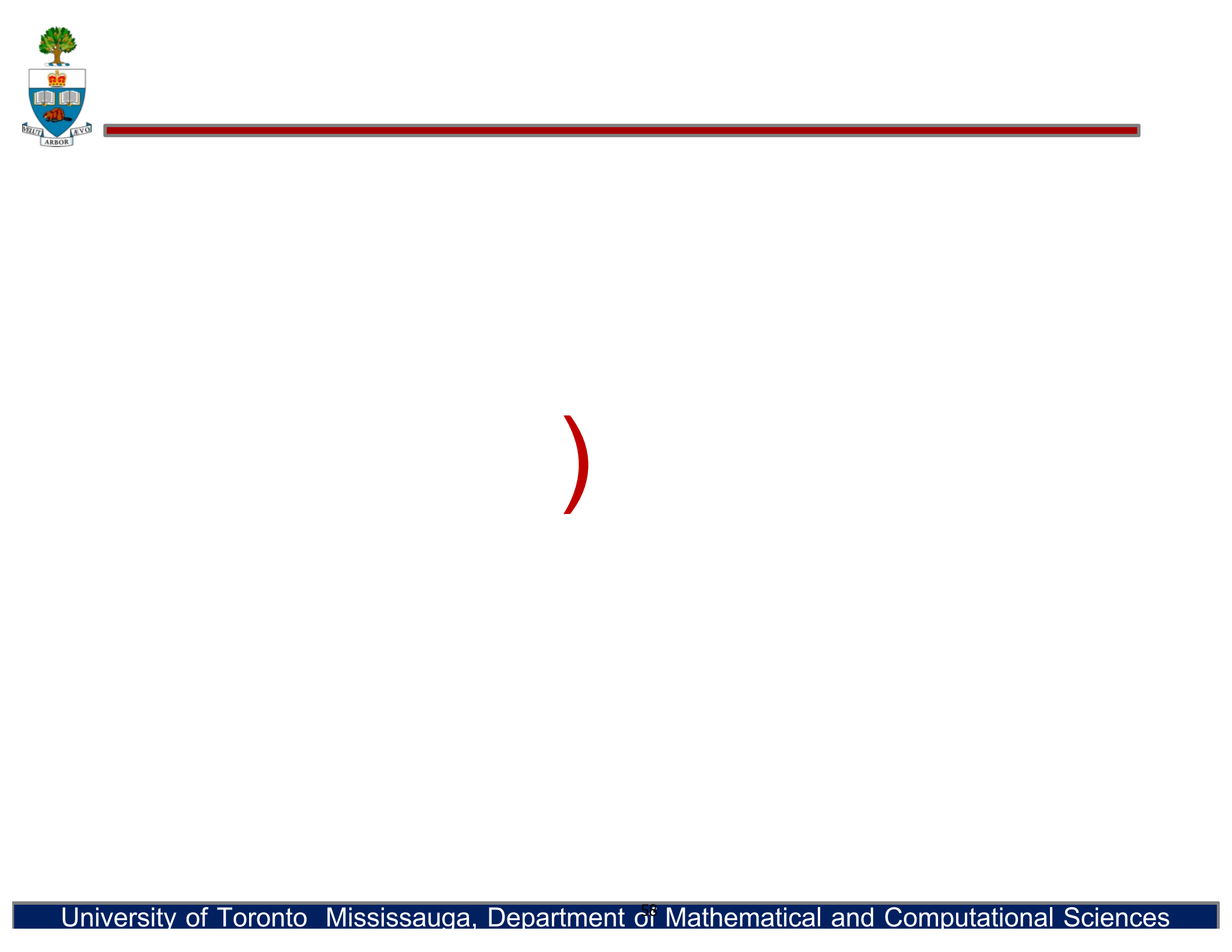
)



2


VELUT
ARBOR
ÆVO

b



(a * ((3) + (b))) / (2 - b)

Third expression

- Ready?

2

7

2 +)7(

Fourth Expression

- We'll give you this one in one go
- Take the same approach
- Ready?

(a+((b-(c/(d*e))-(f+g))))

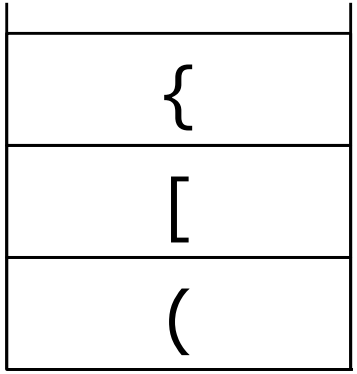
- Once you're done, go to Q2
  - Think, discuss with your group...

Check for “balancedness”

- Why do it like this?
- The computer only “sees” one character at a time

(1 + [2 * { 9 / 3 }] – 4)

- Stack can be used to check balanced parentheses as we've seen in the previous examples

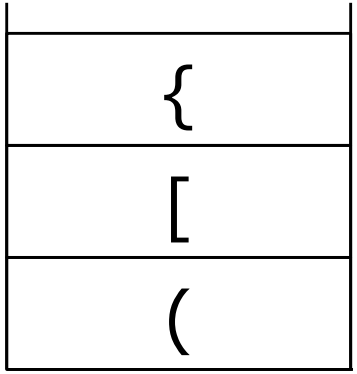


Check for “balancedness”

- Let’s see what happens if imbalanced:

(1 + [2 * { 9 / 3 }) – 4]

- Stack can be used to check balanced parentheses
  - Push when it’s a “leftie”, pop and compare when it’s a “rightie”



What other balance errors can be detected?

Implementation

- Now design your function!
  - Discuss with your group, collaborate, share ideas