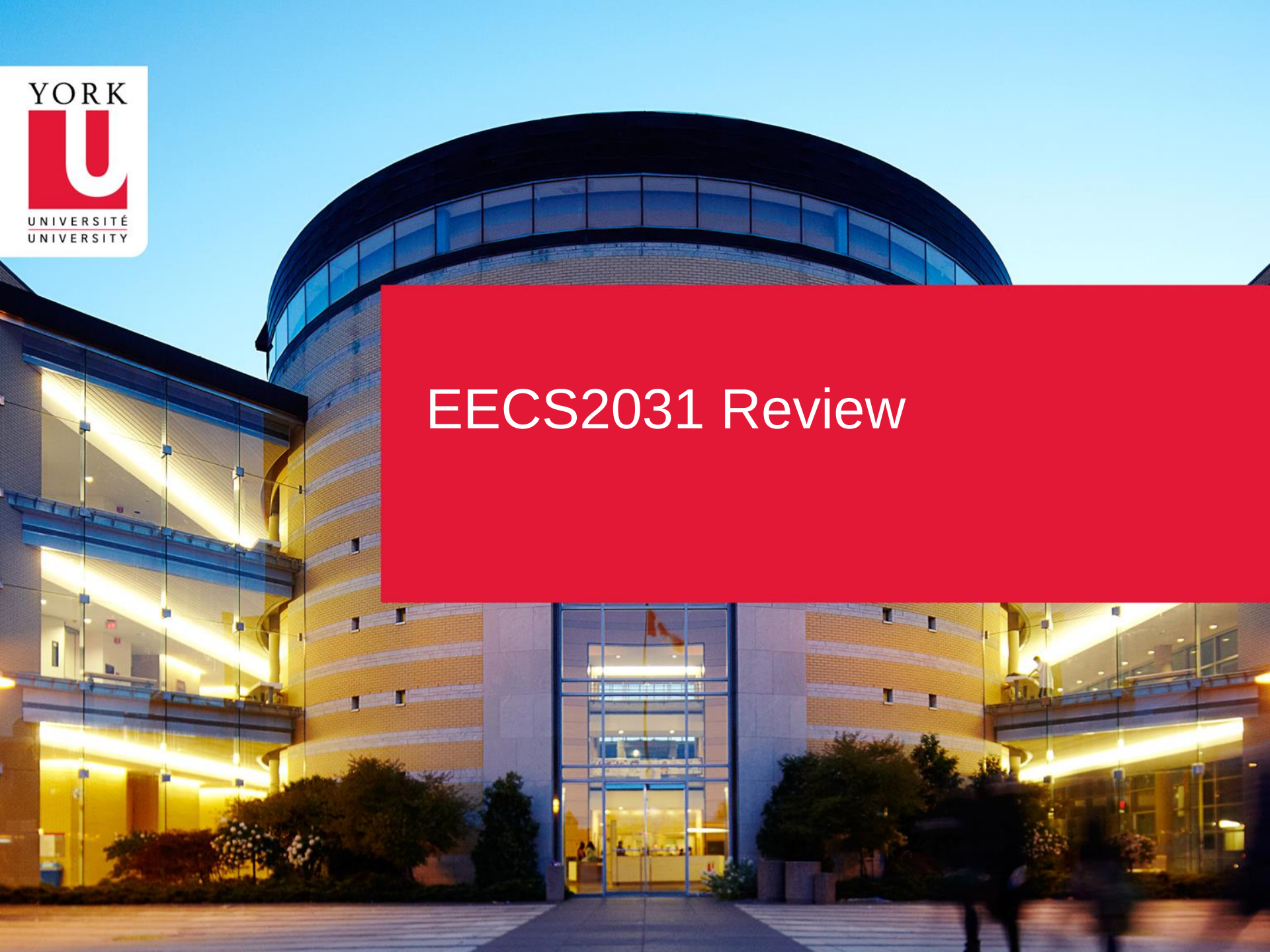


EECS2031 Review

EECS2031: Software tools …

- Unix
  - files and directories
  - permissions
  - utilities/commands
- Shell
  - programming
  - quoting
  - wild cards
  - files

... and programming in C

- C
  - basic syntax
  - types
  - arrays
  - pointers
  - functions
  - strings
  - structs
  - header files

Course Topics

- Topics covered
  - Part I: Unix, Shell & Shell Programming
  - Part II: Programming in C
  - Part III: UNIX System Programming

Part I

Unix, Shell & Shell Programming

Unix Philosophy

- Write programs that do one thing well
- Write programs that work together
- Write programs to handle text streams because that is the universal interface

File interface

- “Everything is a file”
- We treat all sorts of devices as if they were files, and use the file interface (open, read, write, close) all over the place
  - files
  - directories
  - pipes
  - sockets
  - kernel info via /proc (interface to kernel data structures)

Shell Concepts

- Stdin, stdout, stderr
- I/O redirection (>, <, …)
- Process control (ps, kill,…)
- Job control (fg, bg, %, …)
- Pipes (|)

Bourne shell programming

- quoting
  - single quotes (' … ') inhibit wildcard replacement, variable substitution and command substitution
  - double quotes (" … ") inhibit wildcard replacement only
  - back quotes (` … `)cause command substitution
- variables – environment and local
  - `str1="string"`
  - `str2="string"`
  - `if test $str1 = $str2; then … fi`

Bourne shell programming

- `test -f filename` – test if a file exists
- Command line arguments
  - $0 = name of script, $1 .. $n = arguments
- `set` assigns positional parameters to a list of words
- `read` – reads from stdin
- `expr` – math functions

Compiler vs. Interpreter

- Compiler translates whole program to object code
  - produces the most highly optimized code
- Interpreter translates one line of code at a time
  - can quickly make changes and try things out
- C – compiled
- Java – compiled to byte code, then interpreted
- Shell –interpreted

Part II

Programming in C

Programming in C

- Memory model
  - pointers are addresses with a type
- Remember that no variables are automatically initialized
- Arrays
  - contiguous region of memory with fixed size
  - provide random access
- Pointers
  - dereference with *
  - get the address of a variable with &

Strings

- Arrays of characters
- Remember the null termination character ('\0')
- Most string functions depend on it
- Whenever possible use the string functions rather than re-implementing them
- E.g., use `strncpy` rather than copying each character
- Be careful to ensure that you don't walk off the end of a character array

Dynamic memory allocation

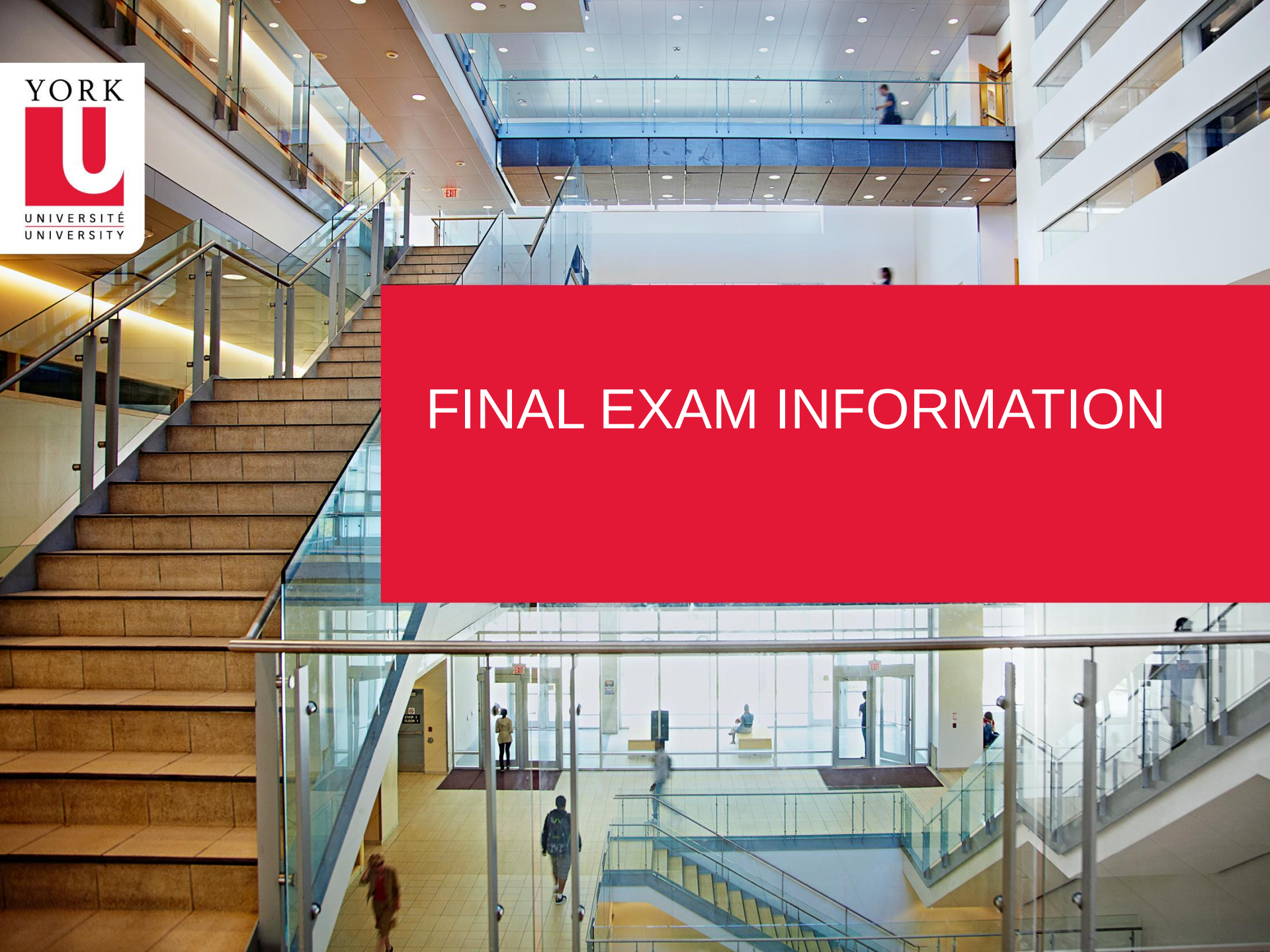
- `malloc, calloc, realloc`
- memory allocated using `malloc` should be freed when it is no longer needed
- keep a pointer to the beginning of the region so that it is possible to free
- **memory leak** occurs when you no longer have a pointer to a region of dynamically allocated memory

When to use malloc?

- When passing a pointer to a new region of memory back from a function
- When you don't know until runtime how much space you need

Makefile & Header files

- Header files contain function prototypes and type definitions
- Header files are useful when your program is divided into multiple files
- Use Makefile to compile programs. Saves typing and takes advantage of separate compilation


YORK
UNIVERSITÉ
UNIVERSITY
FINAL EXAM INFORMATION

Final Exam

- Course Material & How to study
  - Textbook
  - Lecture slides
  - Tutorials – play with example code provided
  - Assignments – make sure you understand concepts and code
- Covers everything in the course
- Closed book exam – No Aids Allowed

Remainder

- Pre-exam Office Hours:
  - Thu, Apr 5 @ 10:30-11:30pm (LAS3050)
  - Or, I'll post in the discussion board about holding an office hour in a classroom (Wed or Thu)
- Final Exam

  WHEN: Mon, Apr 9 @ 2-5pm

  WHERE: TM TAIT
- Course Evaluations ☺

Happy Holidays 2018

YORK UNIVERSITÉ UNIVERSITY