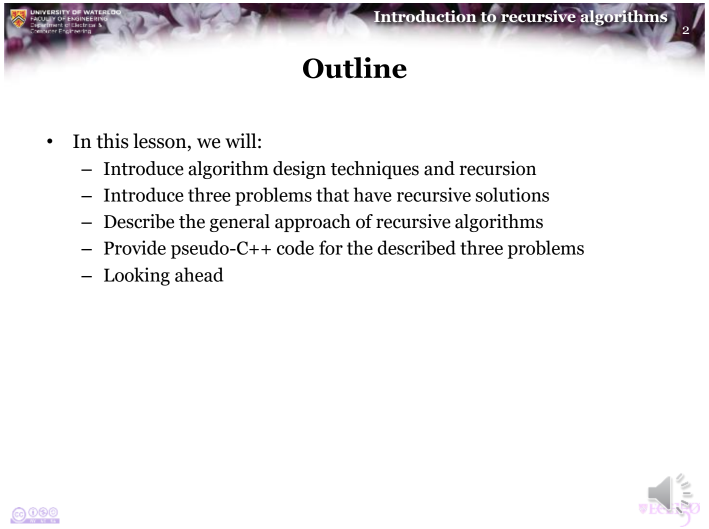
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*

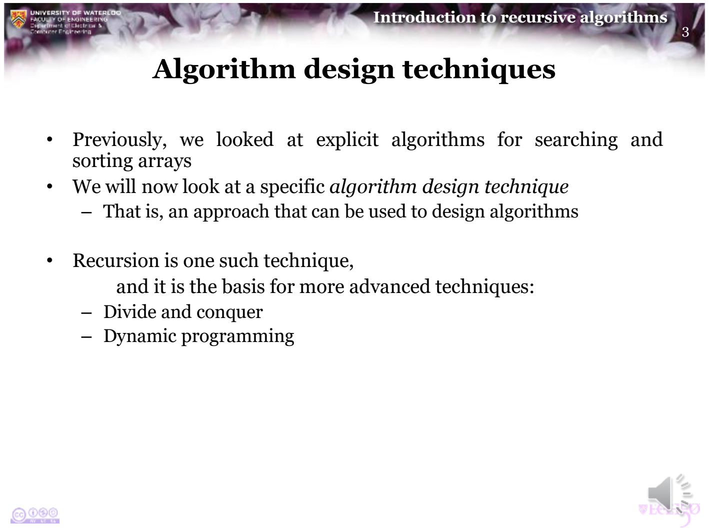
Introduction to recursive algorithms

Douglas Wilhelm Harder, M.Math., LEL
Prof. Hiren Patel, Ph.D., P.Eng.
Prof. Werner Dietl, Ph.D.



Outline

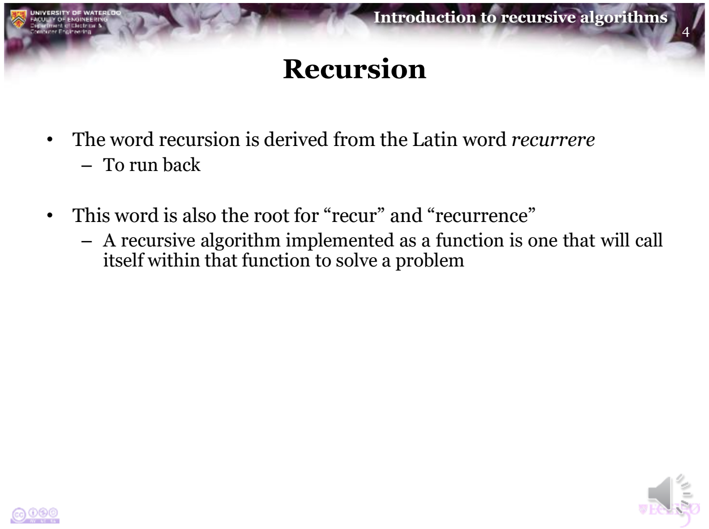
- In this lesson, we will:
  - Introduce algorithm design techniques and recursion
  - Introduce three problems that have recursive solutions
  - Describe the general approach of recursive algorithms
  - Provide pseudo-C++ code for the described three problems
  - Looking ahead



Algorithm design techniques

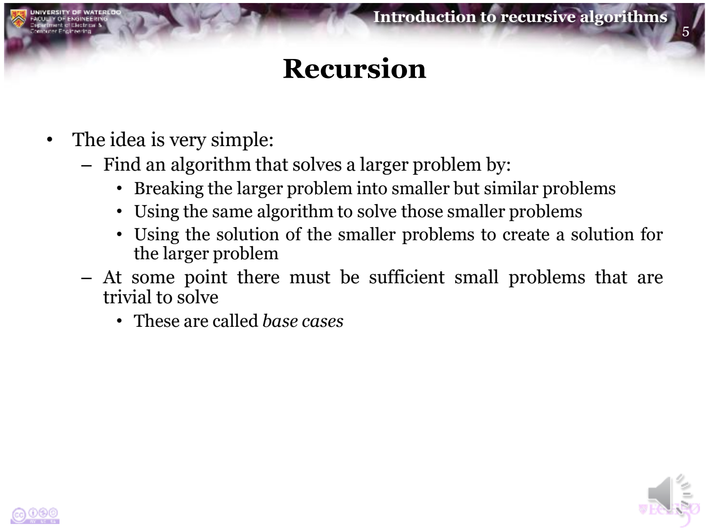
- Previously, we looked at explicit algorithms for searching and sorting arrays
- We will now look at a specific *algorithm design technique*
  - That is, an approach that can be used to design algorithms
- Recursion is one such technique,

  and it is the basis for more advanced techniques:
  - Divide and conquer
  - Dynamic programming



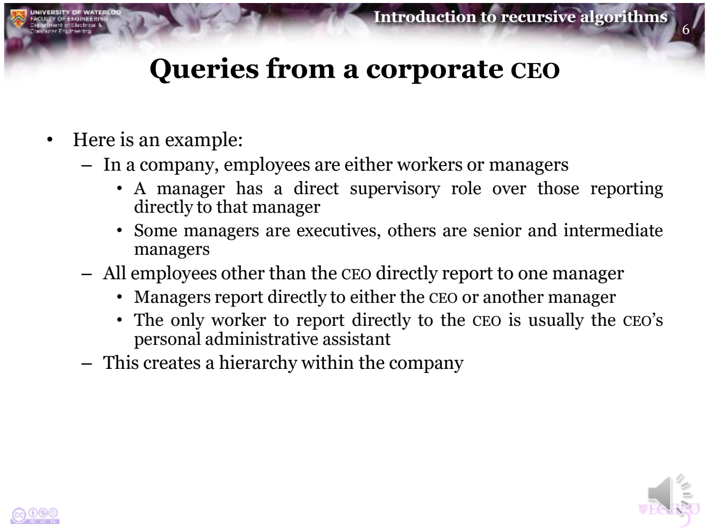
Recursion

- The word recursion is derived from the Latin word *recurrere*
  - To run back
- This word is also the root for “recur” and “recurrence”
  - A recursive algorithm implemented as a function is one that will call itself within that function to solve a problem



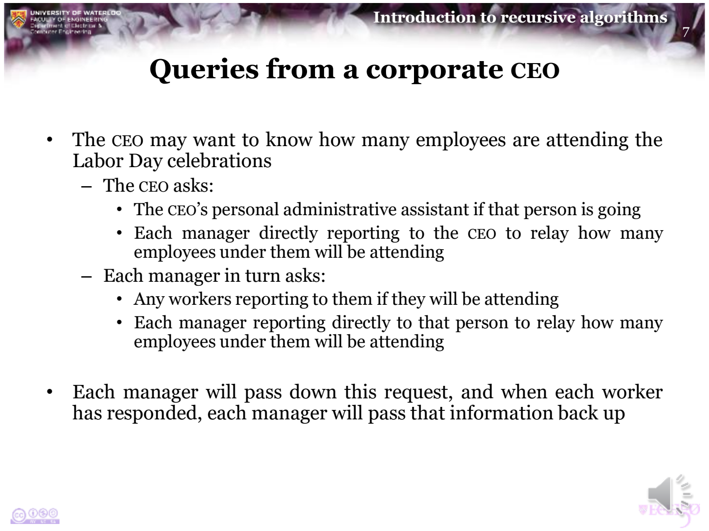
Recursion

- The idea is very simple:
  - Find an algorithm that solves a larger problem by:
    - Breaking the larger problem into smaller but similar problems
    - Using the same algorithm to solve those smaller problems
    - Using the solution of the smaller problems to create a solution for the larger problem
  - At some point there must be sufficient small problems that are trivial to solve
    - These are called *base cases*



Queries from a corporate CEO

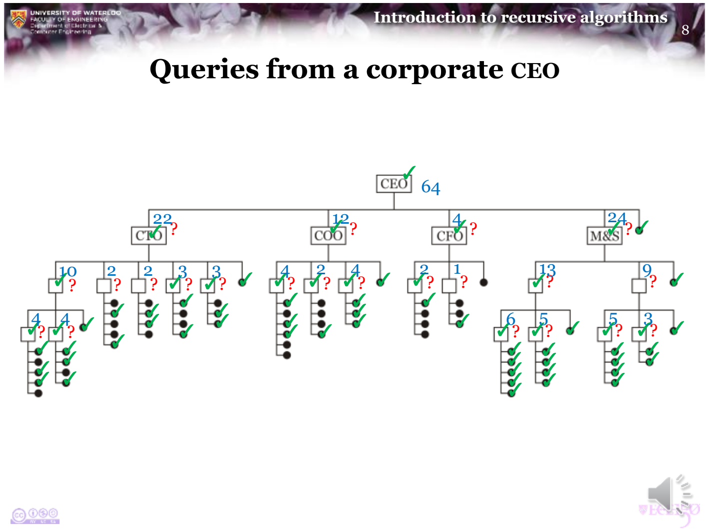
- Here is an example:
  - In a company, employees are either workers or managers
    - A manager has a direct supervisory role over those reporting directly to that manager
    - Some managers are executives, others are senior and intermediate managers
  - All employees other than the CEO directly report to one manager
    - Managers report directly to either the CEO or another manager
    - The only worker to report directly to the CEO is usually the CEO's personal administrative assistant
  - This creates a hierarchy within the company



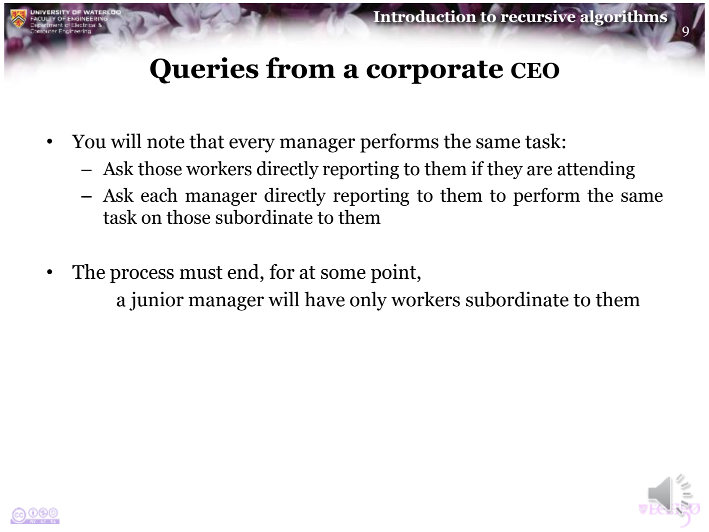
Queries from a corporate CEO

- The CEO may want to know how many employees are attending the Labor Day celebrations
  - The CEO asks:
    - The CEO's personal administrative assistant if that person is going
    - Each manager directly reporting to the CEO to relay how many employees under them will be attending
  - Each manager in turn asks:
    - Any workers reporting to them if they will be attending
    - Each manager reporting directly to that person to relay how many employees under them will be attending

- Each manager will pass down this request, and when each worker has responded, each manager will pass that information back up

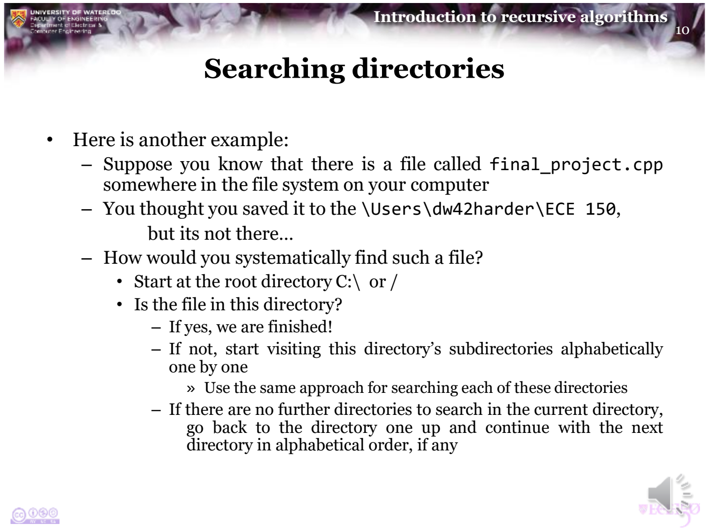


Queries from a corporate CEO



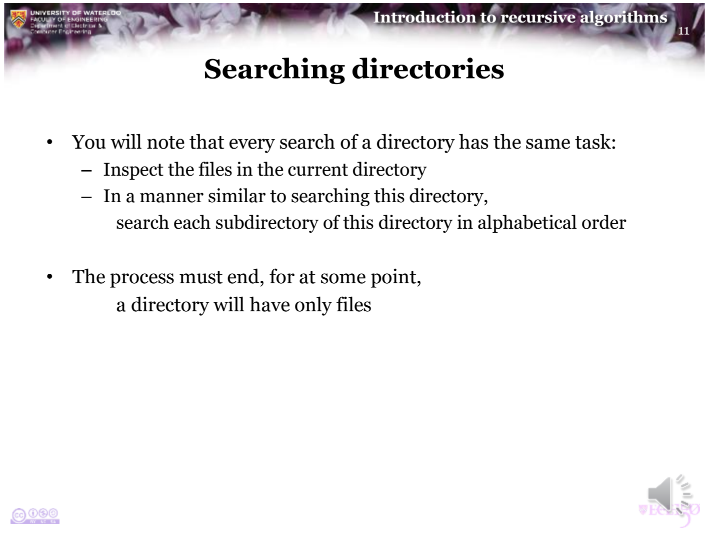
Queries from a corporate CEO

- You will note that every manager performs the same task:
  - Ask those workers directly reporting to them if they are attending
  - Ask each manager directly reporting to them to perform the same task on those subordinate to them
- The process must end, for at some point,
  a junior manager will have only workers subordinate to them



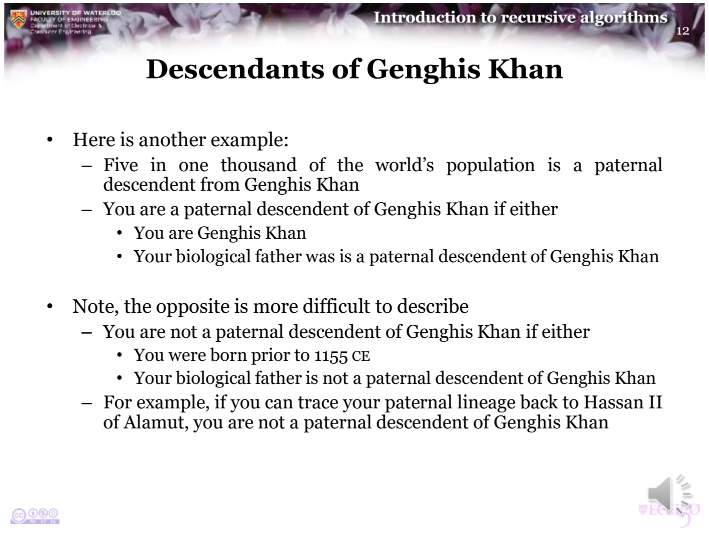
Searching directories

- Here is another example:
  - Suppose you know that there is a file called `final_project.cpp` somewhere in the file system on your computer
  - You thought you saved it to the `\Users\dw42harder\ECE 150`,
    but its not there…
  - How would you systematically find such a file?
    - Start at the root directory C:\ or /
    - Is the file in this directory?
      - If yes, we are finished!
      - If not, start visiting this directory's subdirectories alphabetically one by one
        - » Use the same approach for searching each of these directories
      - If there are no further directories to search in the current directory, go back to the directory one up and continue with the next directory in alphabetical order, if any



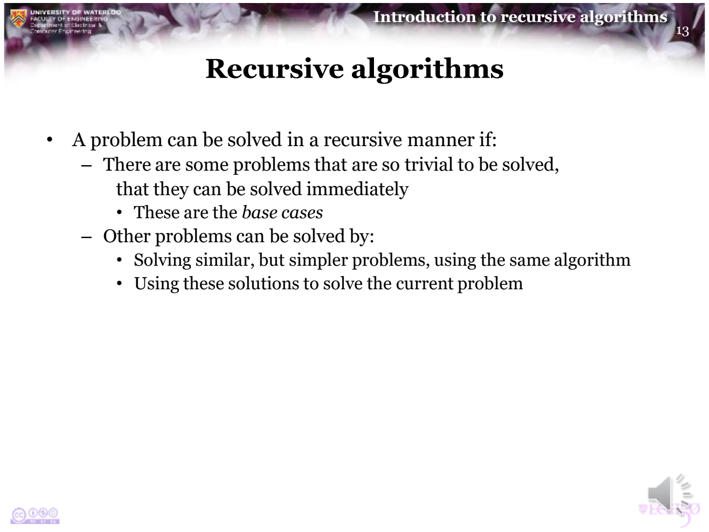
Searching directories

- You will note that every search of a directory has the same task:
  - Inspect the files in the current directory
  - In a manner similar to searching this directory,
    search each subdirectory of this directory in alphabetical order
- The process must end, for at some point,
  a directory will have only files



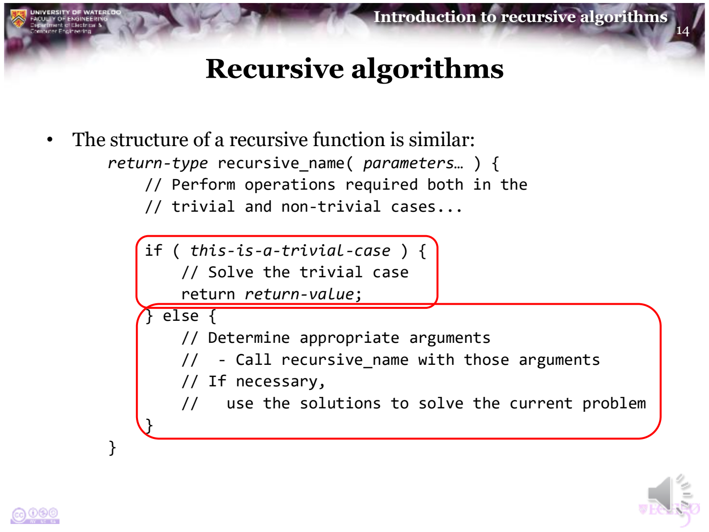
Descendants of Genghis Khan

- Here is another example:
  - Five in one thousand of the world's population is a paternal descendent from Genghis Khan
  - You are a paternal descendent of Genghis Khan if either
    - You are Genghis Khan
    - Your biological father was is a paternal descendent of Genghis Khan
- Note, the opposite is more difficult to describe
  - You are not a paternal descendent of Genghis Khan if either
    - You were born prior to 1155 CE
    - Your biological father is not a paternal descendent of Genghis Khan
  - For example, if you can trace your paternal lineage back to Hassan II of Alamut, you are not a paternal descendent of Genghis Khan



Recursive algorithms

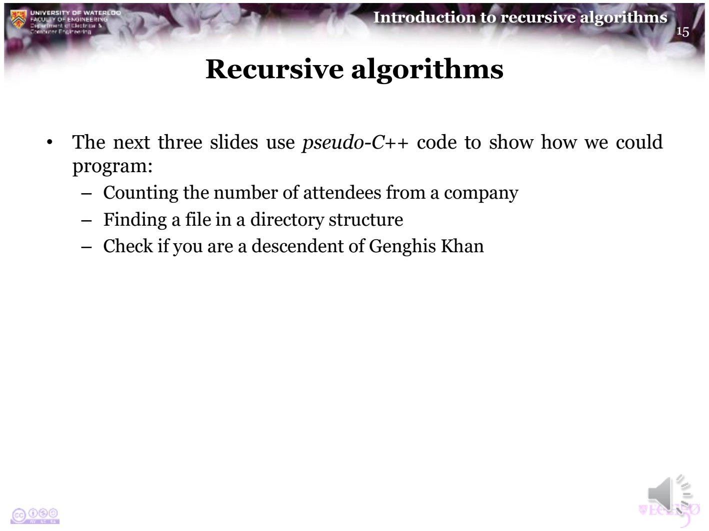
- A problem can be solved in a recursive manner if:
  - There are some problems that are so trivial to be solved, that they can be solved immediately
    - These are the *base cases*
  - Other problems can be solved by:
    - Solving similar, but simpler problems, using the same algorithm
    - Using these solutions to solve the current problem



Recursive algorithms

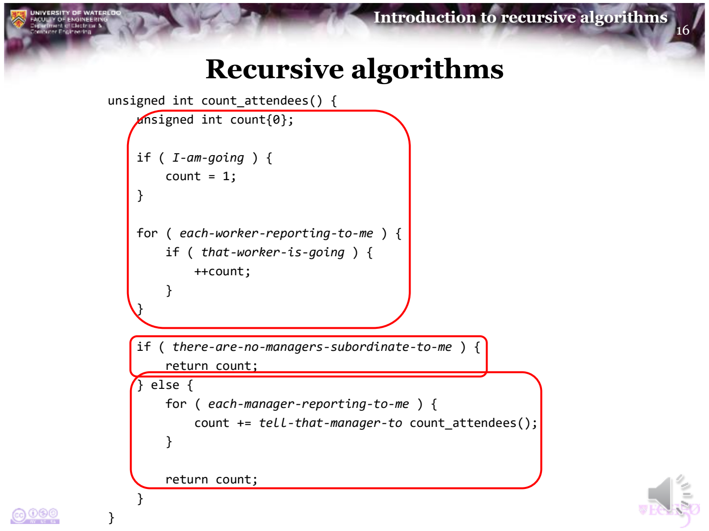
- The structure of a recursive function is similar:

```
return-type recursive_name( parameters… ) {
    // Perform operations required both in the
    // trivial and non-trivial cases...

    if ( this-is-a-trivial-case ) {
        // Solve the trivial case
        return return-value;
    } else {
        // Determine appropriate arguments
        //  - Call recursive_name with those arguments
        // If necessary,
        //   use the solutions to solve the current problem
    }
}
```



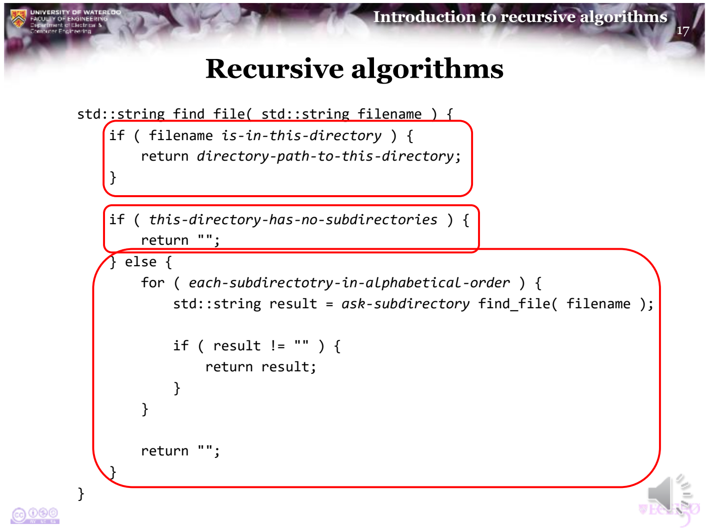
Recursive algorithms

- The next three slides use *pseudo-C++* code to show how we could program:
  - Counting the number of attendees from a company
  - Finding a file in a directory structure
  - Check if you are a descendent of Genghis Khan



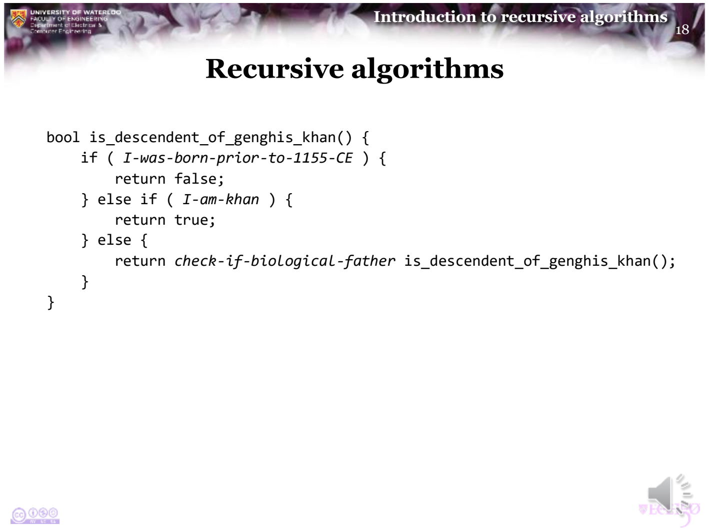
Recursive algorithms

```
unsigned int count_attendees() {
    unsigned int count{0};

    if ( I-am-going ) {
        count = 1;
    }

    for ( each-worker-reporting-to-me ) {
        if ( that-worker-is-going ) {
            ++count;
        }
    }

    if ( there-are-no-managers-subordinate-to-me ) {
        return count;
    } else {
        for ( each-manager-reporting-to-me ) {
            count += tell-that-manager-to count_attendees();
        }

        return count;
    }
}
```



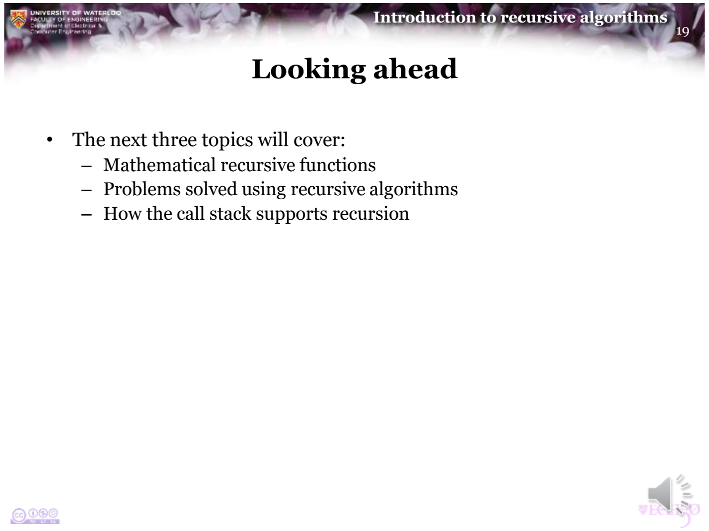
Recursive algorithms

```
std::string find_file( std::string filename ) {
    if ( filename is-in-this-directory ) {
        return directory-path-to-this-directory;
    }

    if ( this-directory-has-no-subdirectories ) {
        return "";
    } else {
        for ( each-subdirectotry-in-alphabetical-order ) {
            std::string result = ask-subdirectory find_file( filename );

            if ( result != "" ) {
                return result;
            }
        }

        return "";
    }
}
```



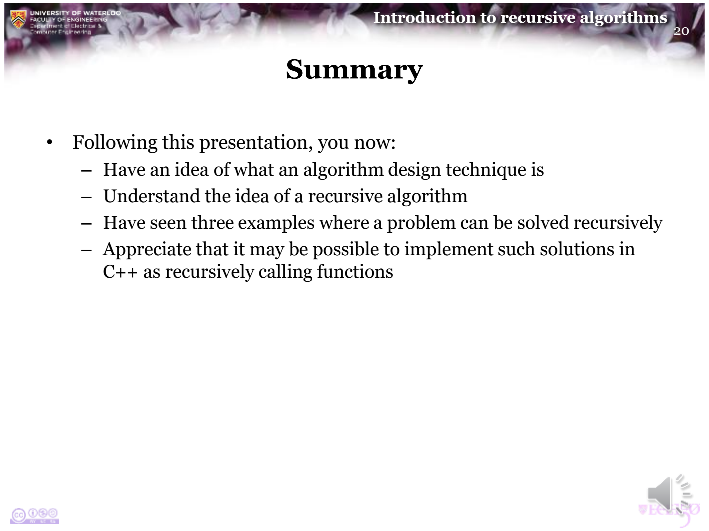
Recursive algorithms

```
bool is_descendent_of_genghis_khan() {
    if ( I-was-born-prior-to-1155-CE ) {
        return false;
    } else if ( I-am-khan ) {
        return true;
    } else {
        return check-if-biological-father is_descendent_of_genghis_khan();
    }
}
```



Looking ahead

- The next three topics will cover:
  - Mathematical recursive functions
  - Problems solved using recursive algorithms
  - How the call stack supports recursion



Summary

- Following this presentation, you now:
  - Have an idea of what an algorithm design technique is
  - Understand the idea of a recursive algorithm
  - Have seen three examples where a problem can be solved recursively
  - Appreciate that it may be possible to implement such solutions in C++ as recursively calling functions

References

[1] Wikipedia,
https://en.wikipedia.org/wiki/Recursion_(computer_science)

[2] Dictionary of Algorithms and Data Structures (DADS)
https://xlinux.nist.gov/dads/HTML/recursion.html

Acknowledgments

None so far.

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.