CPSC 131 – Data Structures

Sequence Container Checkpoint Review

Professor T. L. Bettens

Fall 2020

Overview

How many Questions?	**About 5 to 10 questions, but each question may have many, many parts**
What kind of questions?	Pulldown selection, multiple choice, true or false, short answer, etc.
Will I have to write code?	Yes. Write code by selecting code fragments or filling in the blank.
Will I have to draw data structures?	Yes. Draw data structures after each step in a sequence by pulldown selection or fill in the blank
Will I have to write recursive functions?	Yes, you should be able to solve a problem either iteratively or recursively. You may be asked to do both.
How long do I have?	You have two attempts up to 60 minutes each to complete. I'll take the highest of the two.

What do I need to know?

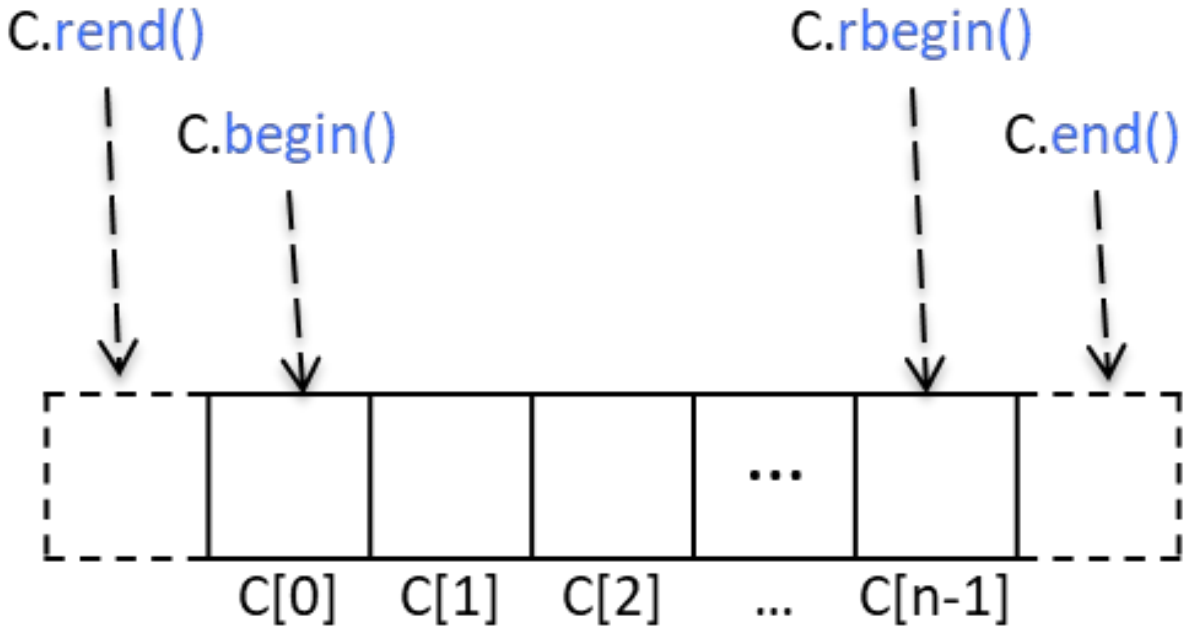
- Short answer, everything
  - Everything we've talked about
  - Everything in the book
  - Everything in Implementation Example code

Okay, What else do I need to know?

- Recursive functions
  - Public function
    - Called by the client.
    - This function calls the overloaded private recursive function with a starting value
  - Private function
    - Base case – how is the recursion terminated?
    - Pre-order vs post-order traversal – do you visit the node then recurse? Or recurse then visit the node?
    - Recurse – call yourself with the next term (the next node in a list for example)

Will I need to know about iterators?

- Yes, the basic ones
  - begin(), rbegin()
  - end(), rend()



Will I need to know the complexity of STL container operations?

- Yes – you may be asked to identify the order of complexity for operations on
  - std::array<T, N>
  - std::vector<T>
  - std::forward_list<T>
  - std::list<T>

Will I need to know more than one way to implement a linked list?

- Yes – be able to code an insert, remove, or find function for
  - singly linked list
    - Null terminated
    - Two dummy nodes
  - Doubly linked list
    - Null terminated
    - Two dummy nodes
    - One dummy node in a circular list

Example: Singly linked list recursive find function

SLinkedList.hpp

```
template <typename Data_t>
class SLinkedList
{
  public:
    // Client visible interface (the public function)
    Iterator find( const Data_t & data );

    ...

  private:
    // The private helper function
    Iterator find( const Data_t & data, Iterator current );

    ...
};
```

SLinkedList.hxx

```
// Client visible interface (the public function)
template<typename Data_t>
SLinkedList<Data_t>::Iterator SLinkedList<Data_t>::find( const Data_t & data )
{
  return find( data, begin() );
}

// The private helper function
template<typename Data_t>
SLinkedList<Data_t>::Iterator SLinkedList<Data_t>::find( const Data_t & data, Iterator current )
{
  if(   current == end() ) return nullptr;
  if( *current == data  ) return current;

  return find( data, current.next() );
}
```

See Sequence Container Implementation Examples