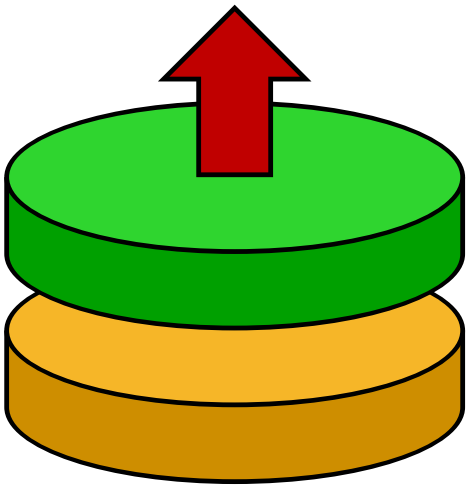
California State University, Sacramento
College of Engineering and Computer Science

Computer Science 130: Data Structures and Algorithm Analysis

Fall 2025 – Project #2 – Proper Stack & Queue

Overview

For this assignment, you are going to use your well-written Linked List to create a nice, and efficient, Stack and Queue. Make sure to do a good job on these classes, you will use them in a future assignment.



Part 1: Generic Key-Value Class

Before we continue, we need to make a simple modification to your Linked List. This will include a new function to remove the item from the head.

But wait, the Node Class (which is hidden deep inside the LinkedList class) is set to **private**. Are we going to make it public now? No. You never want to expose the secret internal working of a data structure. So, we will use a helper class.

This class won't have any functionality except to hold a couple pieces of data. The class definition is below:

```
class KeyValue
   public String key
   public String value
end class
```

When you write this class, it should be in it's own .java file (or .cpp if you use C++).

Part 2: Modifying Your Node Class

Node Class – Modified		
	Node(String)	Constructor. You should split the string at the first colon and use this to assign key and value.
	Node(String key, String value)	NEW Constructor
Node	**Next**	The next node in the chain.
String	**Key**	Key value
String	**Value**	The value / description of the item.

Part 2: Modifying Your Linked List Class

Before you create your Stack and Queue, you need to make a simple modification to your LinkedList class. In particular, we need to be able to remove from the head. Also, let's use your new KeyValue class.

Modify the following methods in your LinkedList Class. If the user passes in a string, we will split in at the colon automatically.

LinkedList Class – Modified Functions		
void	**AddHead(String value)**	Split the string at the first colon. You can then create a new KeyValue instance and call the AddHead(KeyValue) method.
void	**AddTail(String value)**	Split the string at the first colon. You can then create a new KeyValue instance and call the AddTail(KeyValue) method.
String	**ToList()**	You will have to tweak this code slightly.

Add the following functions to your class.

LinkedList Class – NEW Functions		
void	**AddHead(KeyValue item)**	It might be a good idea to modify your existing AddHead method().
void	**AddTail(KeyValue item)**	It might be a good idea to modify your existing AddTail method().
KeyValue	**RemoveHead()**	Removes a node from the head of the list and returns the two values in a generic KeyValue instance. I have the pseudocode listed below.

When you create your Stack and Queue, you want to keep the LinkedList class private (and hidden) from the client (the user of your class). So, an instance of the LinkedList could be created inside both your Stack and Queue. Naturally, this should be private.

Part 3: Your Stack & Queue

The Stack Class

The following is the interface (public functions) for the Stack Class. This will be fairly easy to write – since it wraps around your LinkedList class. In other words, these functions merely call the appropriate function on the instance of the LinkedList. **Don't inherit**.

class Stack		
void	**Push(KeyValue item)**	Pushes a KeyValue onto the stack.
KeyValue	**Pop()**	Pops (removes) a KeyValue from the top of the stack.
boolean	**IsEmpty()**	Returns the value from the internal linked-list class.
String	**ToList()**	Return the string from the internal linked-list class.

The Queue Class

Like before, these functions should call the appropriate function on the private LinkedList instance. **Don't inherit**.

class Queue		
void	**Enqueue(KeyValue item)**	Enqueues a string onto the queue.
KeyValue	**Dequeue()**	Dequeues (removes) a string from the front of the queue.
boolean	**IsEmpty()**	Returns the value from the internal linked-list class.
String	**ToList()**	Return the string from the internal linked-list class.

Part 4: Testing

File Format

A number of test files will be provided to you for testing your code. The format is designed to be easy to read in multiple programming languages. You need to use the classes, built in your programming language, to read the source files.

```
Key: Value 1
Key: Value 2
...
Key: Value n
```

The following is one of the most basic test files on the website.

halloween calories.txt

```
73: M&M's Fun size
60: Tootsie Pop
40: Starburst Fun Size
70: Kit Kat Snack Size
30: Laffy Taffy
80: Snickers Fun Size
50: Nerds Mini Box
77: Hershey's Milk Chocolate Fun Size
82: Almond Joy Snack Size
```

How to Test

1. Read the input file into an instance of your Stack and Queue
2. Print them to the screen to verify they were loaded correctly.
3. Write a loop and dequeue KeyValue objects from the Queue and print them to the screen
4. Write a loop and pop all the KeyValue objects from the Stack and print them to the screen.
5. Afterwards, try manually adding a few KeyValues to the Queue and Stack see if you get pop/dequeue them

Allowed Programming Languages

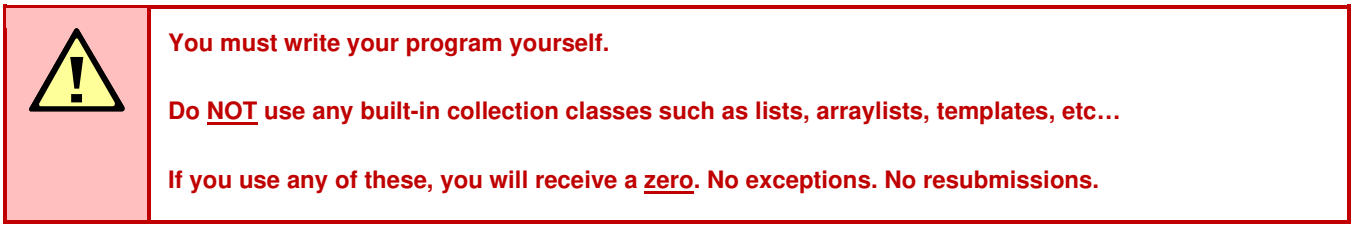
You may use any of the following programming languages.

- C#
- C++ (not recommended)
- Java

The following **cannot** be used:

- C
- Groovy
- JavaScript
- Kotlin
- Lua
- Nim
- Python
- Ruby
- Scala
- Swift
- TypeScript
- Visual Basic .NET

Requirements



The following are the requirements for this assignment:

- This **must** be completely all your code. If you share your solution with another student or re-use code from another class, you will receive a zero.
- Do not use any built-in Stack or Queue class (many programming languages provide them).
- You **must** encapsulate (i.e. make a private instance) of your LinkedList in both the Stack and Queue. **Do not inherit.**
- You must use your LinkedList from Project 1.
- You may use any of the programming languages listed below.
- Create some excellent testing for your class.
- Proper style (see below).
- Do not put your `main()` method inside the LinkedList class. This will cause major issues going forward.

Grading

1	New LinkedList functions	20%
2	The KeyValue class	15%
3	LinkedList class is encapsulated (private) in Stack and Queue	20%
4	Correct interface for Queue and Stack	10%
5	Proper output	10%
6	Proper Style	10%
7	Testing by Reading the File	15%

Due Date

Due **October 17, 2025** by **11:59 pm.**

Given you already have developed excellent programming skills in CSc 20, this shouldn't be a difficult assignment.

- **Do not send it to canvas. I will not read nor grade Canvas e-mails.**
- **Do not send a cloud link. I cannot open cloud links.**

E-Mail the following to **dcook@csus.edu**:

- The source code for the classes. Just send the source code files (.java, .cs, .cpp,, etc….)

The e-mail server will delete all attachments that have file extensions it deems dangerous. This includes .jar, .exe, .class, and many more.

So, only send the following types of files:

- `.txt`
- `.java`
- `.cs`
- `.cpp`
- `.h`

Please send a ZIP File containing all your files.

Some Helpful Pseudocode

RemoveHead Pseudocode

```
function RemoveHead() returns KeyValue
    if the list is empty                          ... is head null?
        result = null                             ... We could also throw an error
    else if (this.head == this.tail)              ... Just 1 node. Set head/tail to null
        result = new KeyValue using the head's values
        this.head = null                          ... Deference both
        this.tail = null
    else                                          ... 2 or more nodes.
        result = new KeyValue using the head's values
        this.head = this.head next;               ... Link new node to the current head
    end if

    return result
end function
```