**COURSE TITLE : DATA STRUCTURES AND ALGORITHMS**

**INSTRUCTOR : MR. MARK KENNETH LIMJOCO**

**1. Array Data Structure**

* **Key Concept**

An **array** is a fundamental data structure that stores a collection of elements of the same data type in contiguous memory locations. Each element in an array is identified by an index, and you can access elements by their index.

* **Learning Tasks**
* **A screenshot of a computer

  Description automatically generatedMerging four arrays into one.**
* **A screenshot of a computer screen

  Description automatically generatedDelete element in an array.**
* **A screenshot of a computer program

  Description automatically generatedSearch element in an array.**
* **Takeaways**
* Arrays are collections of elements stored in contiguous memory locations, accessible by index.
* Merging arrays involves copying elements from multiple arrays into a single array.
* Deleting an element from an array may require shifting elements to fill the gap.
* Searching for an element in an array involves iterating through the array and comparing each element until a match is found or the end is reached.

**2. LinkList Data Structure**

* **Key Concept**

A **linked list** is a linear data structure in which elements are stored in nodes, and each node points to the next node in the sequence. Linked lists consist of a head (the first node) and a tail (the last node). They are dynamic in size and efficient for insertions and deletions.

* **Learning Tasks**
* **A screenshot of a computer

  Description automatically generatedMerging of three LinkList into one**.

**A screenshot of a computer

Description automatically generated**

* **A screenshot of a computer

  Description automatically generatedA screenshot of a computer

  Description automatically generatedDelete node at the beginning and end.A screenshot of a computer

  Description automatically generated**
* **A screenshot of a computer program

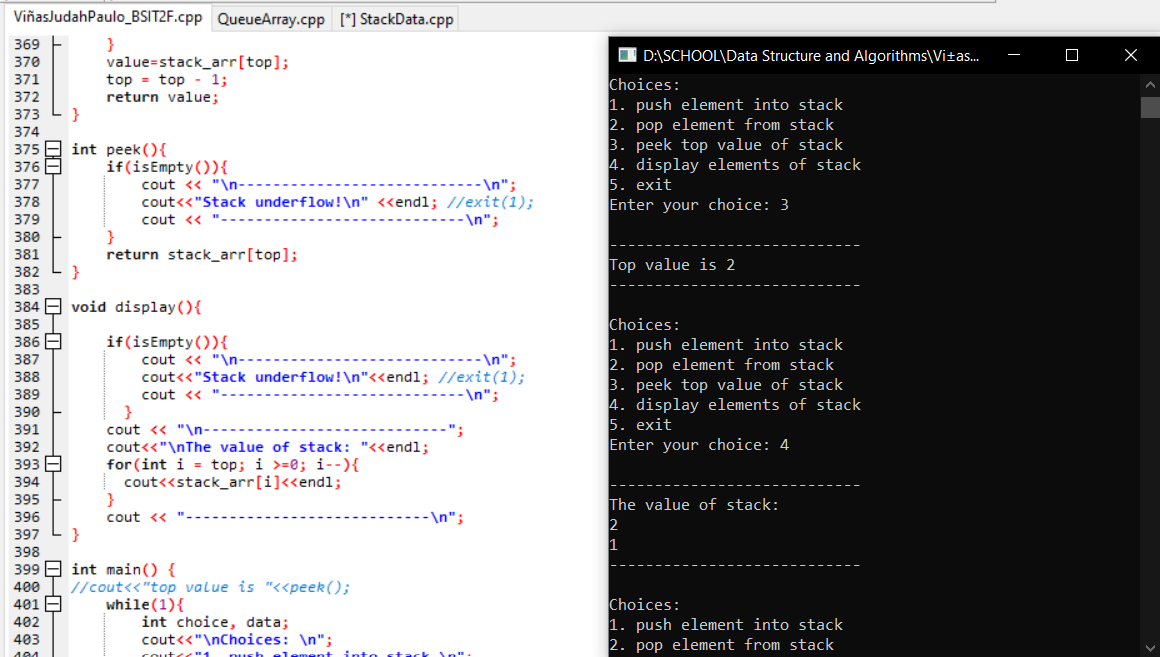
  Description automatically generatedA screenshot of a computer

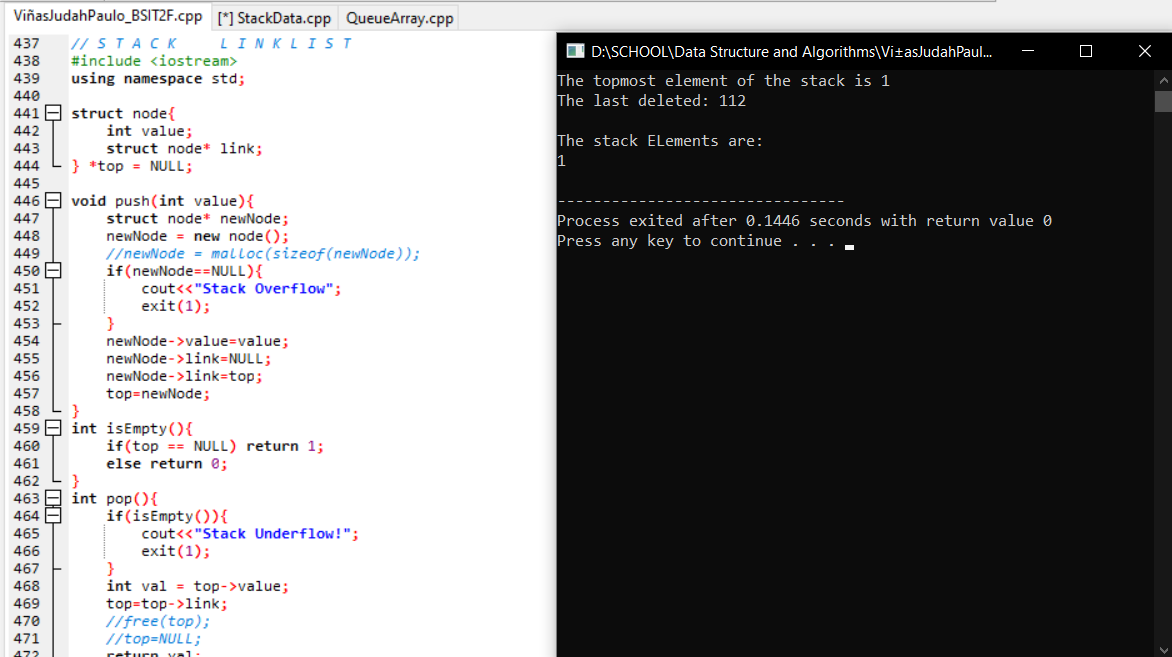
  Description automatically generatedInsert node at the beginning, middle and end.**
* **Takeaways**
* Linked lists are dynamic data structures where elements are stored in nodes, and each node has a pointer to the next node.
* Merging linked lists involves updating the next pointers appropriately.
* Deleting and inserting nodes at different positions in a linked list require updating pointers accordingly.
* Linked lists are useful when you need dynamic size and efficient insertions and deletions compared to arrays.

**3. Stack Data Structure**

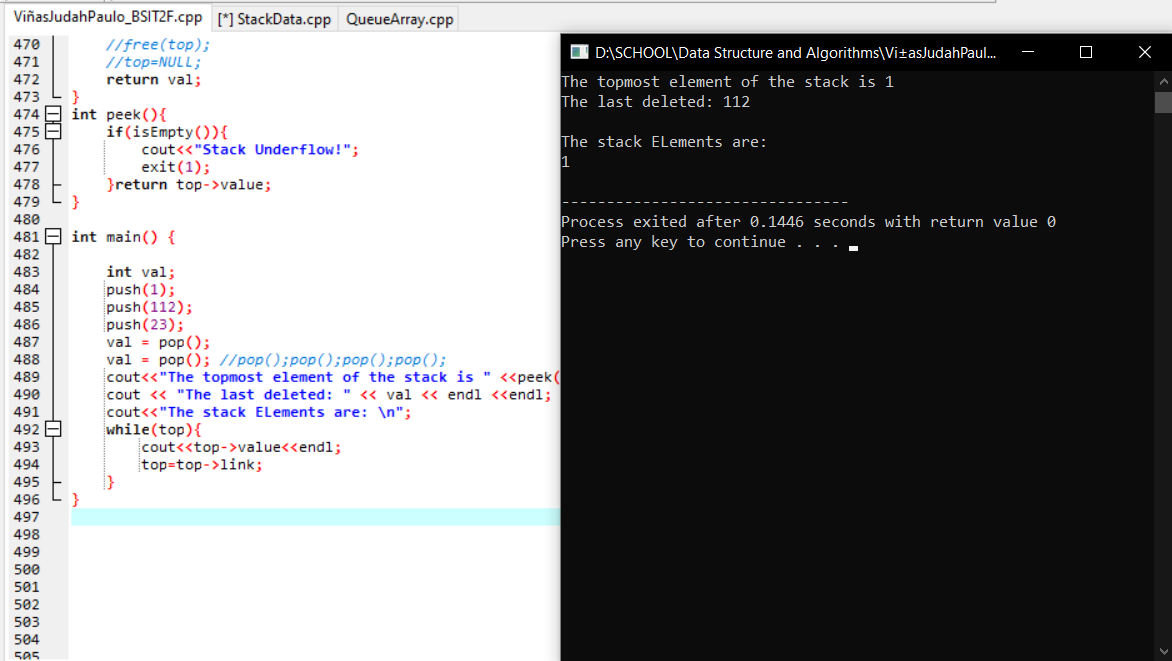
* **Key Concept**

A **stack** is a linear data structure that follows the Last-In-First-Out (LIFO) principle. In a stack, elements are added and removed from the same end, known as the top. It is often used for tasks that involve keeping track of the order of elements, such as function call execution and expression evaluation.

* **Learning Tasks**
* **A screenshot of a computer program

  Description automatically generatedStack Array**
* A screenshot of a computer program

  Description automatically generated**Stack LinkList**

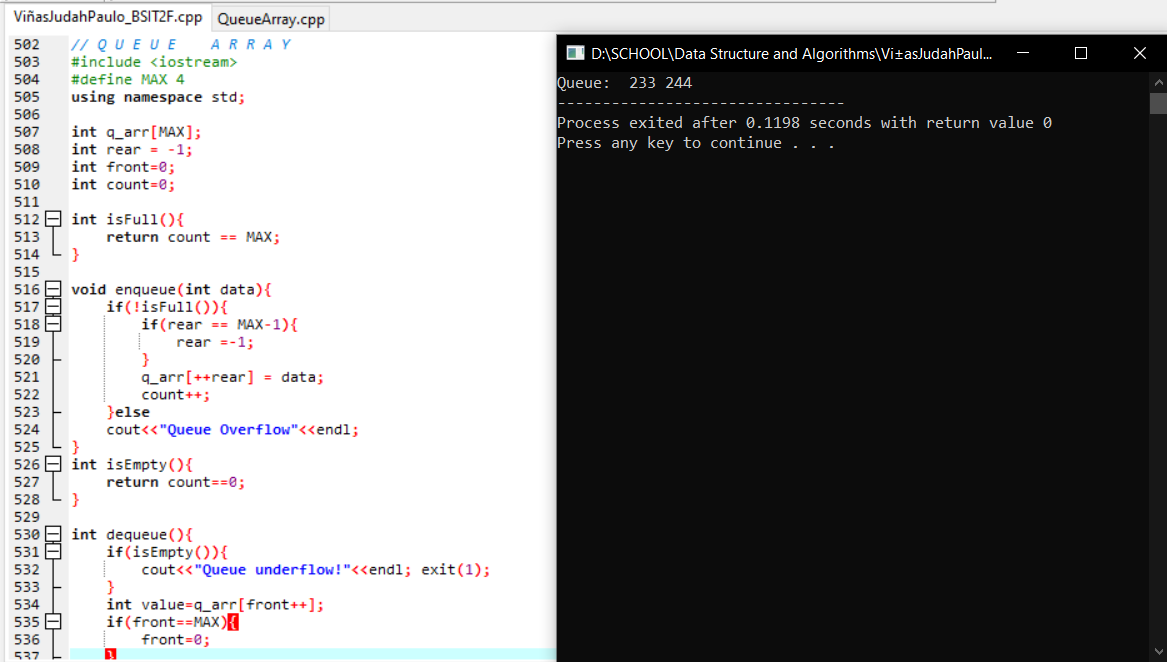
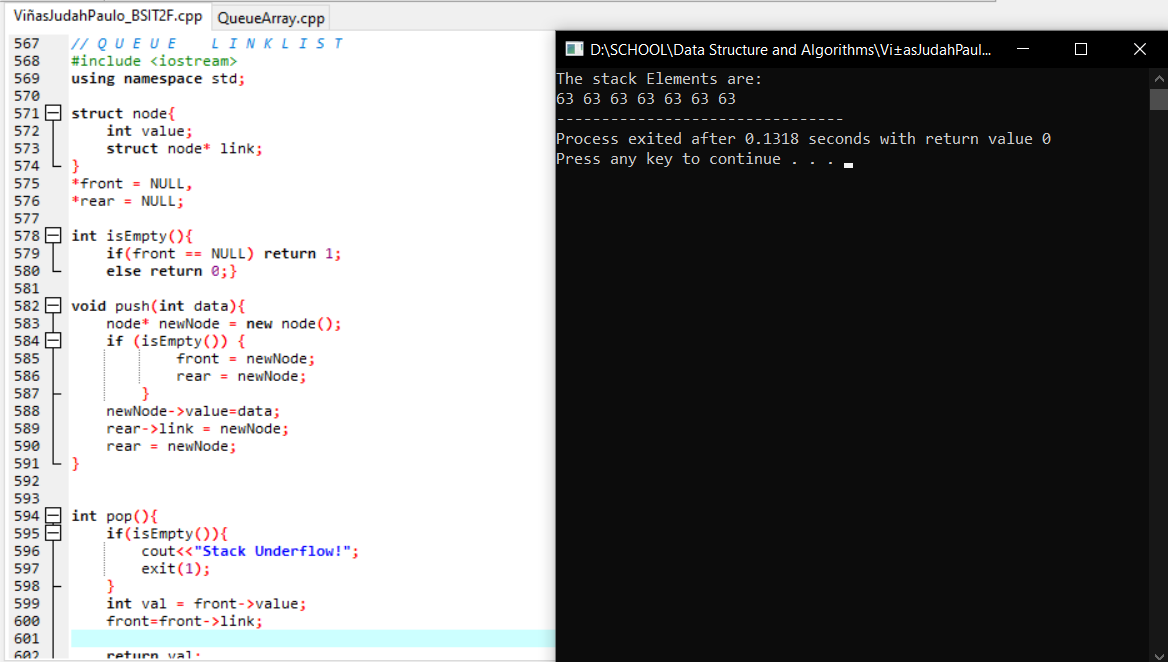
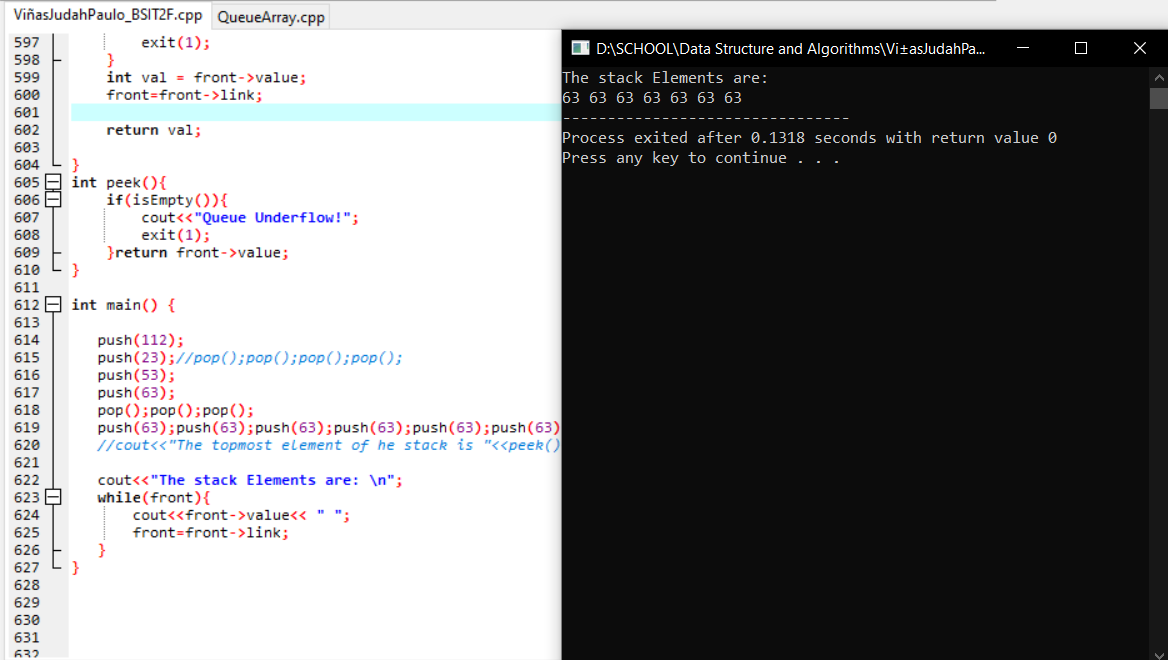


* **Takeaways**
* A stack is a data structure that follows the LIFO (Last-In-First-Out) principle.
* Elements are added and removed from the top of the stack.
* The ‘stack’ container in C++ Standard Library simplifies stack operations.
* Stacks are commonly used in scenarios where the order of operations matters, such as function call execution, undo functionality, and expression evaluation.

**4. Queue Data Structure**

* **Key Concept**

A **queue** is a linear data structure that follows the First-In-First-Out (FIFO) principle. In a queue, elements are added at the rear (enqueue) and removed from the front (dequeue). It is often used in scenarios where elements are processed in the order they are added, like task scheduling, print queue, etc.

* **Learning Tasks**
* **Queue Array**
* **Queue LinkList**
* **Takeaways**
* A queue is a data structure that follows the FIFO (First-In-First-Out) principle.
* Elements are added to the rear (enqueue) and removed from the front (dequeue) of the queue.
* The ‘queue’ container in C++ Standard Library simplifies queue operations.
* Queues are commonly used in scenarios where the order of processing matters, such as task scheduling and managing resources.

**Overall Reflection:**

**Course Impact:**

The data structures course, which covered arrays, linked lists, queues, and stacks, greatly shaped my general understanding and perspective on data structures and algorithms. It has given me a firm understanding of the fundamental concepts, implementation details, and real-world applications of various data structures.

I obtained practical insights into how these data structures function and how they may be utilized for solving various computational problems through hands-on examples and code implementations. This knowledge has improved not just my programming skills but also my problem-solving ability.

The course also emphasized the importance of choosing the right data structure for individual applications, taking into account considerations such as efficiency, memory utilization, and ease of manipulation. It has increased my understanding of the importance of data structures in software development and their function in optimizing algorithmic solutions.

**Recommendations:**

* **Algorithm Analysis**: Include a section on algorithm analysis for a better understanding of time and space complexity in relation to data structures.
* **Problem-Solving Challenges**: Integrate coding challenges and problem-solving exercises that apply data structure concepts to real-world scenarios for practical experience.
* **Visualizations**: Incorporate visual aids and animations to make abstract data structure concepts more accessible and engaging.