LESSON 3

98-361 Software Development Fundamentals

3.1 Understand Application Lifecycle Management

3.2 Interpret Application Specifications

3.3 Understanding Algorithms and Data Structures

MTA Software Fundamentals 3 Test

LESSON 3.1

98-361 Software Development Fundamentals

Understand Application Lifecycle Management

Lesson Overview

Students will understand application lifecycle management (ALM).

In this lesson, you will learn about:

- The phases of ALM
- Software testing

Review Terms

- Application Lifecycle Management (ALM)—a combination of business management practices and software engineering. ALM phases are Envisioning, Designing, Developing, Testing, and Maintenance. Also known as Software Development Life Cycle.
- Test—to check program correctness by trying out various sequences and input values.
- UML—acronym for Unified Modeling Language, a language used for specifying, building, and documenting software and non-software systems, such as business models. UML notation provides a common foundation for object-oriented design by providing descriptions of modeling concepts, including object class, associations, interface, and responsibility.

What is ALM?

- A combination of business management practices and software engineering.
- ALM phases can be split into five phases:
  - Envisioning, Designing, Developing, Testing, and Maintenance
- The objective is to model the full cycle through which software is developed, managed, and deployed.

Benefits of Collaboration within the ALM

- Increases productivity, as the team shares best practices for development and deployment, and developers can focus on current business requirements
- Reduces the number of deficiencies due to miscommunication
- Catches inconsistencies between requirements
- Accelerates development
- Cuts maintenance time by synchronizing application and design
- Increases flexibility by reducing the time it takes to build and adapt applications that support new business initiatives

Envisioning

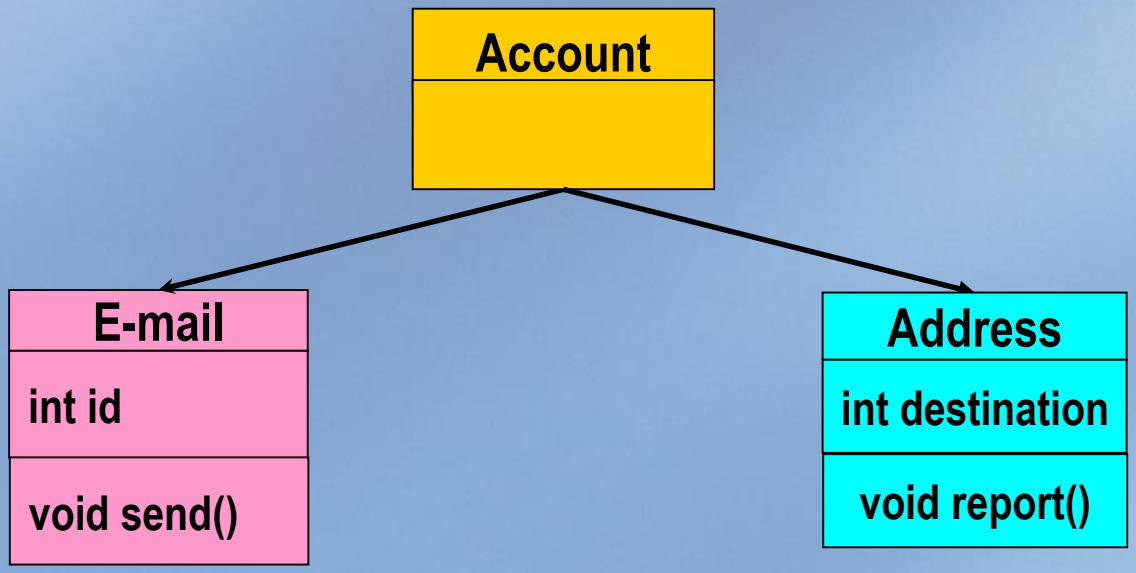
- Customer needs are evaluated and criteria are established to track the project's progress.
- Business case development
  - Costs, benefits, and return on investment are weighed.
- Plans are developed for delivering the product.
  - Breaking the project down into shippable increments
- Processes are developed to measure the quality of the product and the efficiency of the team in developing working software.

Design

- Create models to ensure that the software system meets users' needs.
- The models are developed to different levels of detail and are related to one another, to tests, and to the development plan.
- In the early stages of the project, the overall requirements, design, and tests are developed at an outline level.

Unified Modeling Language (UML)

- UML is used for specifying, building, and documenting software systems.



Development

- Write and document the code.
- Write and perform unit tests.
  - Identify the tests that must be run if you make a particular change.
- Debug and analyze the code.
  - Associate your code changes with specific tasks and bugs.
- Plan and track your progress against your schedule.
- This process is iterative.

Testing

- Unit testing
  - Testing of classes happens separately from the testing of the program as a whole.
- Integration testing
  - Units of software are combined and tested as a group.
- Regression testing
  - Old tests are applied old tests to new versions of the program.

Unit Testing

- A unit test can test a whole class, a group of methods, or even a single method.
- Method stubs help keep test cases independent from one another.
- A test harness can be used to call a method with supplied parameters and compare the results to desired values.
- The benefit is that each individual part of a program can be shown to be correct before they interact with one another.

Integration Testing

- In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested.
- A component, in this sense, refers to an integrated aggregate of more than one unit.
- Many units are combined into components, which are in turn aggregated into even larger parts of the program.
- The idea is to test combinations of pieces and eventually expand the process to test modules with those of other groups.

Regression Testing

- Occurs after a number of failures in previous versions of the software have been discovered and fixed.
- Tests were created as these bugs were found, to verify that the bug fixes really worked.
- These tests are added to a set of tests referred to as a *regression test suite*.
- These tests are collected into a test suite, a set of tests for repeated testing.
- The idea is that when changes are made to create new versions of the program, tests that used to succeed may no longer succeed.

Maintenance

- The deployed application is monitored and managed.
- Updates are made accordingly to cope with new problems (bugs).
- Robust testing will decrease the amount of maintenance.
- New requirements and problems are dealt with separately.
- The new requirements go into the next release while the bug fixes are applied to the current release.

No Student Lab 3.1

LESSON 3.2

98-361 Software Development Fundamentals

Interpret Application Specifications

Lesson Overview

Students will interpret application specifications.

In this lesson, you will learn about:

- Reading and translating specifications into prototypes, code, and components

Review Terms

- Application—a program designed to assist in the performance of a specific task, such as word processing, accounting, or inventory management.
- Component—an individual modular software routine that has been compiled and dynamically linked and is ready to use with other components or programs.
- Database—a collection of tables composed of records, each containing fields together with a set of operations for searching, sorting, recombining, and other functions.
- Service—in reference to programming and software, a program or routine that provides support to other programs.
- Web application—a set of clients and servers that cooperate to provide the solution to a problem.

What is an application specification?

- It describes the technical requirements of an application.
- It can also be specifically targeted at providing the information that developers require to make their application compatible with other applications or systems.
  - Example: When Microsoft Windows Server 2003 was launched, it came with an application specification describing requirements that applications must meet to be certified.
- The application specification describes the problem that needs to be solved and conveys the requirements to the programmer.
- The goal is to provide the programmer with the information required to implement an appropriate solution.

Types of Applications

- Windows service
- Web application
- Web service
- Windows Form application
- Console application
- Database application

Windows Service

- An executable that carries out specific functions and is designed to not require user involvement.
- Windows Service executables often are configured to start alongside the operating system and run in the background.
- Why a Windows Service?
  - When you want a program to start automatically when the operating system starts
  - When your program does not require user interaction, and therefore may not need a user interface
  - When you need long-running functionality

Web Application

- An application accessed using a Web browser
- Usually composed of three tiers:
  - The Web browser (example: Windows Internet Explorer)
  - The Web content engine (example: ASP.NET)
  - The database (example: Microsoft SQL Server)
- Why a Web application rather than a traditional application?
  - Easy to update and maintain
  - Cross-platform compatibility
- Examples: Web mail, online sales

Web service

- A Web service provides the ability to exchange messages in a loosely coupled environment using standard protocols such as Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML).
- A Web service enables the exchange of data and the remote invocation of application logic using XML messaging to move data through firewalls and between heterogeneous systems.
- The only assumption made between the client and the server is that recipients will understand the messages that they receive.

Windows Form Application

- A Windows Form application is a graphical application in which information is displayed and controls are provided to interact with data.
- Why use a Windows Form?
  - The interface corresponds to the operating system, so the application is integrated with the desktop.
  - Consistent user interface.
  - Higher processing demand.
  - Security and reliability is important.
  - Does not require an Internet connection.

Console Application

- A Console application is a computer program designed to be used through a text-only computer interface.
- Why use a Console application?
  - A mouse or pointing device is not required.
  - Speed of deployment.
  - Ease of use.

Database Application

- A database application obtains and manipulates data from a database managed by a database management system (DBMS).
- Typical database applications include programs for data input, data viewing, and batch processing of data.
- Why use a database application?
  - Large amount of data to be stored and retrieved
  - Client/server interaction

No Student Lab 3.2

Microsoft

LESSON 3.3

98-361 Software Development Fundamentals

Understanding Algorithms and Data Structures

LESSON 3.3

98-361 Software Development Fundamentals

Lesson Overview

Students will understand algorithms and data structures

In this lesson, you will learn about:

- Arrays
- Stacks
- Queues
- Linked lists
- Sorting algorithms
- Performance implications of various data structures
- Choosing the right data structure

Review Terms

- array – a list of data values, all of the same type, any element of which can be referenced by an expression consisting of the array name followed by an indexing expression.
- data structure – an organizational scheme, such as a record or array, that can be applied to data to facilitate interpreting the data or performing operations on it.
- linked list – a list of nodes or elements of a data structure connected by pointers.

Review Terms

- queue – a multi-element data structure from which (by strict definition) elements can be removed only in the same order in which they were inserted; that is, it follows a first-in-first-out (FIFO) constraint.
- sort algorithm – an algorithm that puts a collection of data elements into some sequenced order, sometimes based on one or more key values in each element.
- stack – represents a variable size last-in-first-out (LIFO) collection of instances of the same arbitrary type.

What is a data structure?

- Classes used to organize data and provide various operations upon that data
- The type of data structure used for an algorithm can determine its performance
- Choice of data structure is dependent on the context
  - **A stack is best when you want the elements to be accessed in a last-in-first-out order**
  - **A linked list is best when there will be a number of insertions into the data structure**

Arrays

- Arrays are one of the simplest and most widely used data structures in computer programs
- Arrays in any programming language all share a few common properties:
  - **The contents of an array are usually stored in contiguous memory**
  - **All of the elements of an array must be of the same type or of a derived type; hence arrays are referred to as homogeneous data structures**
  - **Array elements can be directly accessed. If you know you want to access the ith element of an array, you can simply use one line of code: `arrayName[i]`**

Arrays

- If a specified array index is out of bounds, an `IndexOutOfRangeException` is thrown
- To change the number of elements an array holds, create a new array instance of the specified size, and then copy the contents of the old array into the new, resized array
- Searching an unsorted array is acceptable when working with small arrays, or when performing very few searches
- If an application stores large arrays that are frequently searched, there are other data structures better suited for the job than arrays

Example: Array code

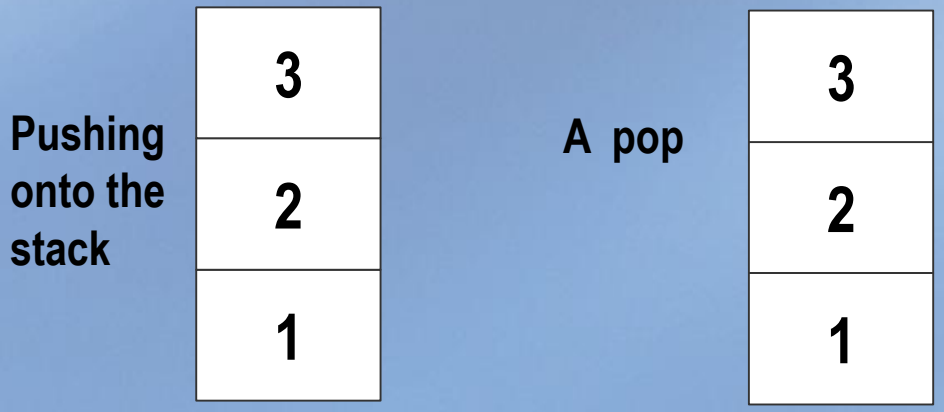
```
public static void Main()


    dim nums() as Integer = new nums(3)
    num(0) = 23
    num(1) = 45
    num)2) = 19
    Console.WriteLine("Index 0: " & nums[0])
    Console.WriteLine("Size: " + nums.Length) for
    (int i = 0, i < nums.length; i++)

        Console.WriteLine(nums[i]);
```

Stacks

- Last-In-First-Out (LIFO) data structure
- Does not allow random access to elements
- Can be visualized graphically as a vertical collection of items
  - When an item is pushed onto the stack, it is placed on top of all other items.
  - Popping an item removes the item from the top of the stack
- Useful when you need access to the item most recently added

Stacks

- The following animation demonstrates items 1, 2, and 3 being pushed onto the stack in that order, and then a single pop.



Example: Stack code

Push(), Pop(), and Count

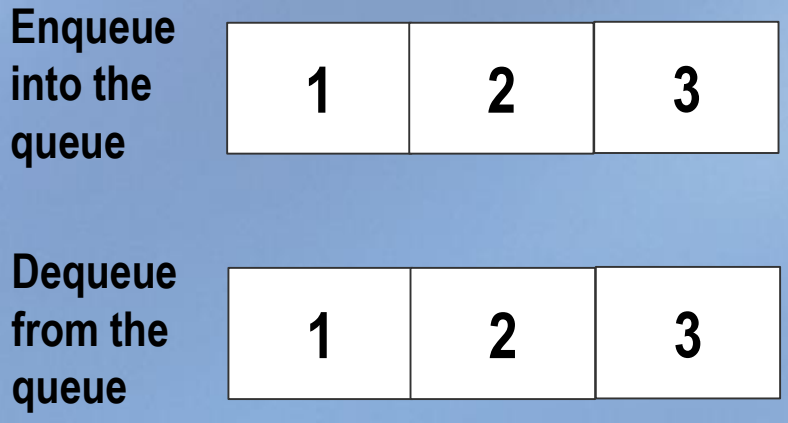
```
public static void Main()

    Stack myStack = new Stack()
    myStack.Push("1")
    myStack.Push("2")
    myStack.Push("3")
    Console.WriteLine("Last in: "+myStack.Pop())
    Console.WriteLine("Elements left:
    "+myStack.Count)
```

Queues

- First-In-First-Out (FIFO) data structure
- Does not allow random access to elements
- Ideal for situations where you are only interested in processing items in the same order in which they were received
- `Enqueue()` and `Dequeue()` are used to input and access elements in the data structure

Queue

- The following animation demonstrates adding items 1, 2, and 3 into the queue (enqueue) and removing an item (dequeue)



Example: Queue code

Enqueue(), Dequeue(), and Count

```
public static void Main()

    Queue myQ = new Queue()
    myQ.Enqueue("1")
    myQ.Enqueue("2")
    myQ.Enqueue("3")
    Console.WriteLine("First in: "+myQ.Dequeue())
    Console.WriteLine("Elements left: "+myQ.Count)
```

Linked Lists

- A list of nodes or elements of a data structure connected by pointers
  - A singly linked list has one pointer in each node pointing to the next node in the list
  - A doubly linked list has two pointers in each node that point to the next and previous nodes
  - In a circular list, the first and last nodes of the list are linked
- Does not allow random access to elements

Linked Lists

- Allow more efficient insertion and removal of elements in the middle of the sequence
  - Because only links are changed, an insertion or removal only affects the neighbors of the inserted or removed element, and does not affect the entire structure
- Insertion and removal is independent of the number of elements in the list, assuming the spot of insertion or deletion is located
- Searching for an element has a linear relationship to the number of elements

Example: Linked List code

```
public static void Main()

    LinkedList<String> myList = new LinkedList
    <String>()
    myList.AddFirst("one")
    myList.AddLast("two")
    Console.WriteLine("First spot:
    "+myList.RemoveFirst())
```

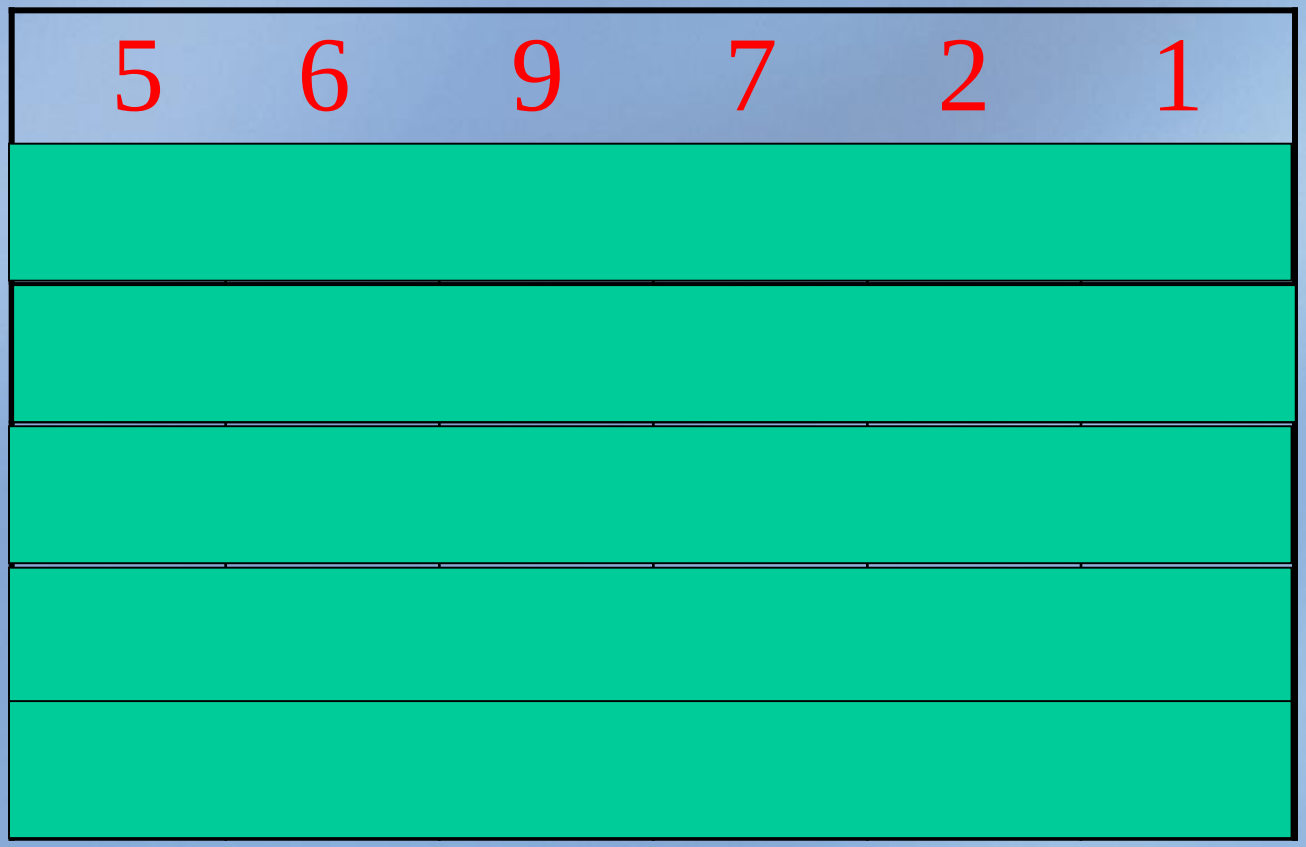
Sorting Algorithms

- Steps that put a collection of data elements into some sequenced order, sometimes based on one or more key values in each element
  - Bubble sort
  - Selection sort
  - Insertion sort

Bubble Sort

- Compares each item in the list with the item next to it, and swapping them if needed
- In one pass, the largest number will eventually end up in the last spot (if the list is being sorted smallest to largest)
- Repeats this process N-1 times, where N is the number of items to sort

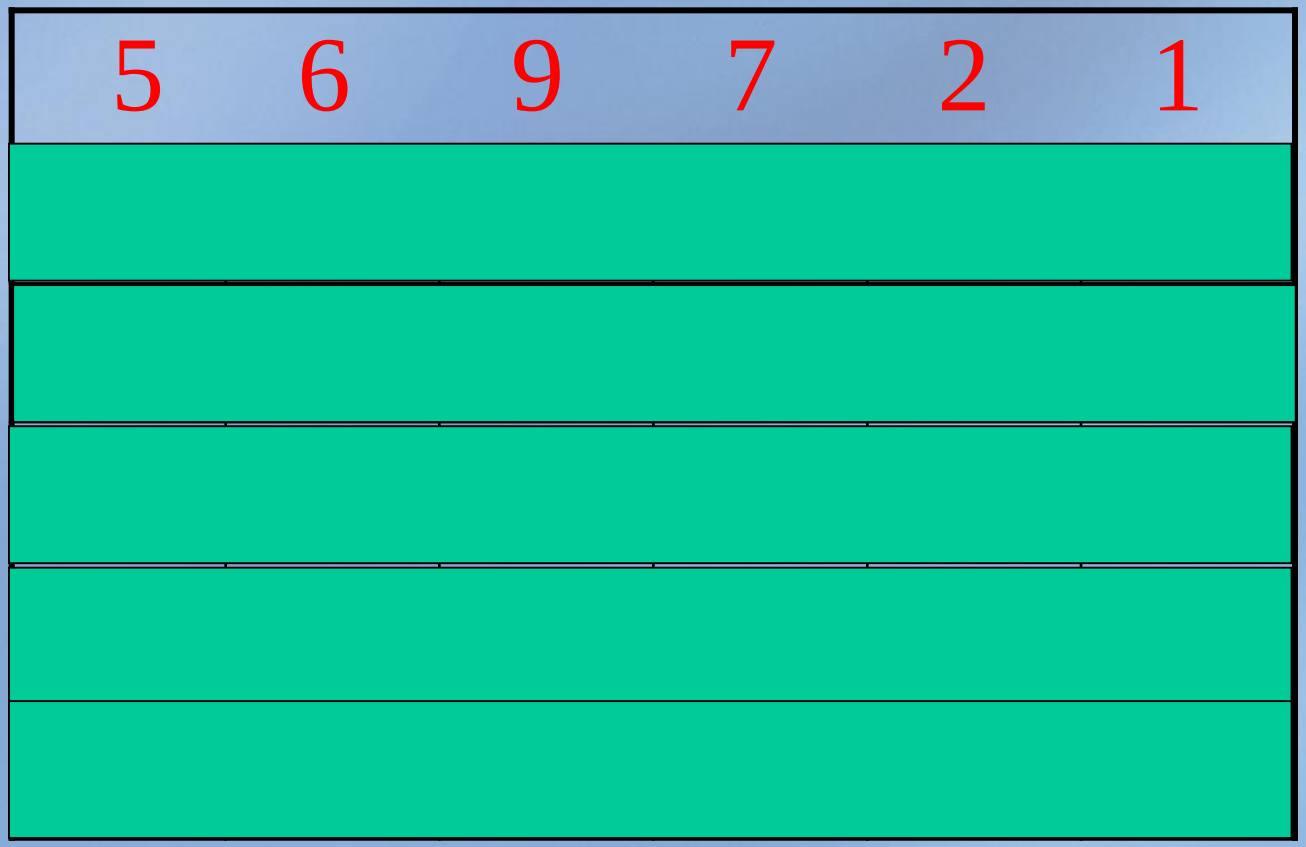
Bubble Sort:



Selection Sort

- Finds the smallest number in the list and swapping it with the number in the index it belongs in
- In each pass, the part of the array that gets scanned for the smallest number decreases
- Repeats this process N-1 times, where N is the number of items to sort

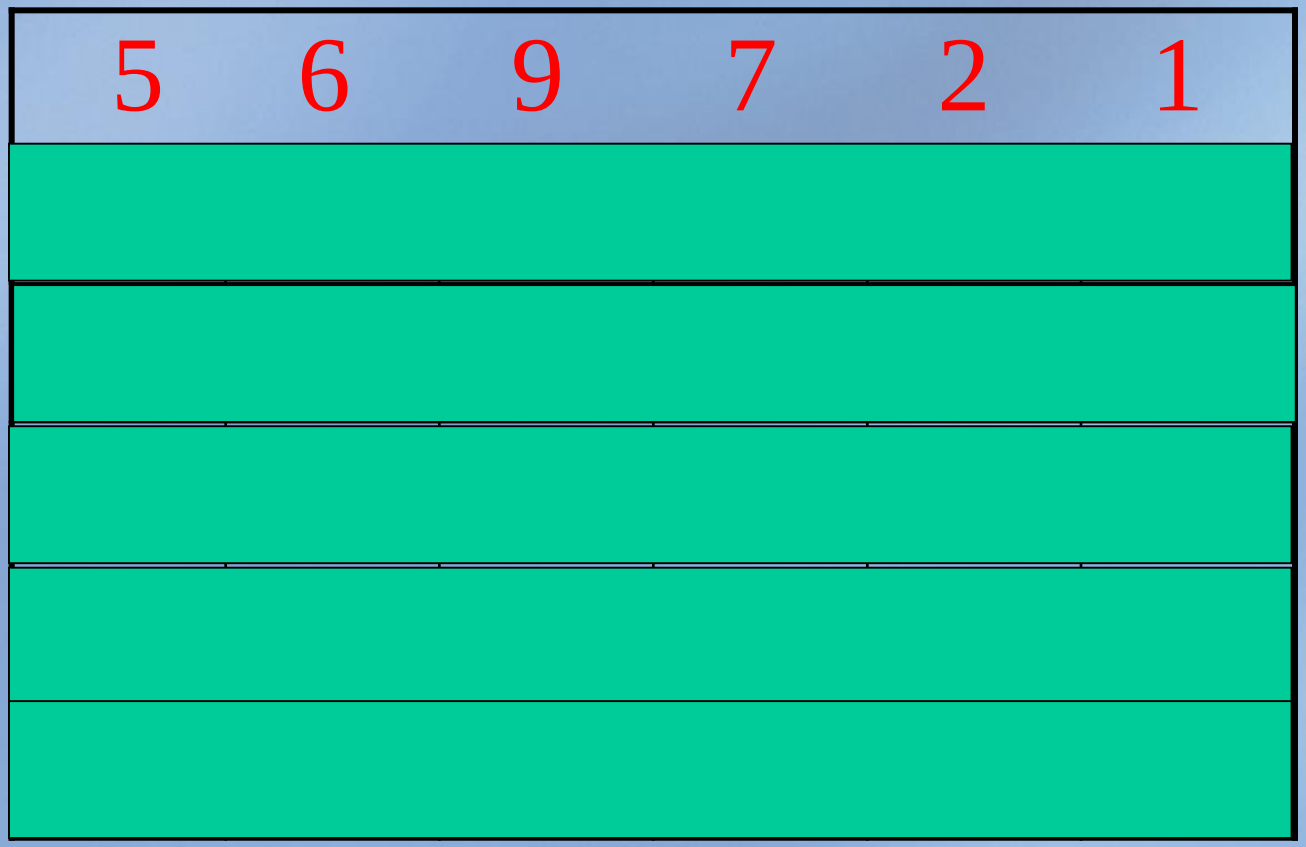
Selection Sort:



Insertion Sort

- The first element in an array is sorted with respect to itself. The array can be said to be composed of a sorted side and an unsorted side
- Elements from the unsorted side are inserted in the correct order into the sorted side one at a time
- To create the insertion spot, elements are moved down to make room
- Repeats this process N-1 times, where N is the number of items to sort

Insertion Sort:



Student Lab 3.3

Complete the QUIA Test

MTA Software Fundamentals 3 Test