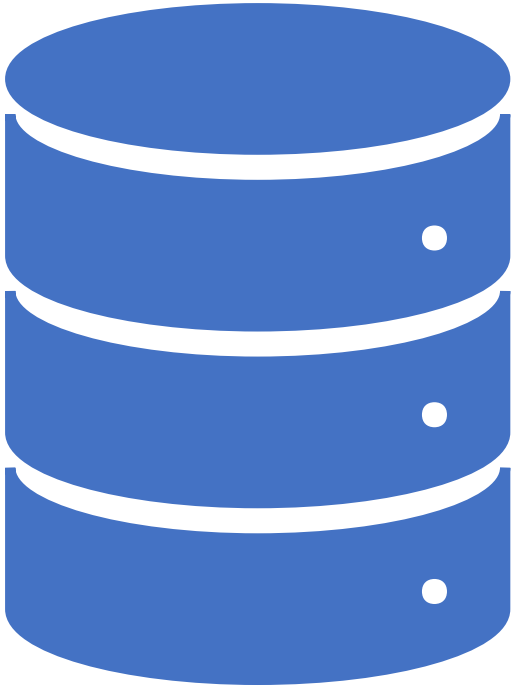


Data Structure

Prepared by Dr.Mohamed

2024

Data Structure

- هياكل البيانات هي أساس تنظيم وتخزين البيانات في علوم الكمبيوتر. وهي تحدد طريقة ترتيب البيانات والوصول إليها عليها.
- "هياكل البيانات" تُستخدم لتنظيم وتخزين البيانات بطريقة مُنظّمة لتسهيل عمليات البحث والإدخال والحذف والتعديل
- هياكل البيانات تحدد العلاقات بين عناصر البيانات والعمليات التي يمكن إجراؤها عليها

A data structure

- A data structure is a way of organizing and storing data so that it can be efficiently accessed and manipulated.
- It defines the relationships between the data elements and the operations that can be performed on them. Here's a simple definition along with an example:

- Definition: A data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

- Example: Let's consider the array data structure.

- Array: An array is a collection of elements of the same data type stored at contiguous memory locations. Each element in the array is identified by its index.

هياكل البيانات Data Structure

- هيكل البيانات هي طريقة محددة لتخزين وتنظيم البيانات في الكمبيوتر؛ وذلك لاستخدامها بسهولة في أداء العمليات البرمجية. حيث أنه لا تصلح عملية التحليل أو العمليات البرمجية الأخرى أن تتم على بيانات غير منظمة.
- تقوم معظم المؤسسات بجمع البيانات وتخزينها بطريقة غير منظمة وغير فعالة في تسهيل استخدام البيانات لاحقًا وهنا تأتى أهمية هياكل البيانات.
- هناك أنواع (أشكال) مختلفة من هياكل البيانات، وكل نوع يستخدم لحل مشكلة معينة، وليس كل المشكلات يتم حلها بنفس هيكل البيانات. لذا، فإن مهمة المبرمج هي فهم بنية البيانات ووضعها في هيكل مناسب لتسهيل عملية تحليل البيانات وجعلها تتم بسرعة وبكفاءة عالية بحيث يمكن استخدامها لاحقًا لحل المشكلات أو تحقيق هدف معين.
- هياكل البيانات هي واحدة من أهم الأساسيات في مجال هندسة البرمجيات وعلوم الكمبيوتر ويتم استخدامها في معظم أنظمة البرمجيات.

Data Structure

تنقسم هياكل البيانات إلى قسمين أساسيين:

- **هياكل بيانات خطية**
- المصفوفات ((Arrays
- القوائم المترابطة ((Linked lists
- الطوابير ((Queues
- الأكوام ((Stacks

هياكل بيانات غير خطية2)

- الشجرة ((Trees
- الرسوم البيانية ((Graphs

Data Structure Type

- Data structures are the foundation of organizing and storing data in computer science. They define the way data is arranged, accessed, and manipulated in computer memory. There are various types of data structures, each with its own advantages and use cases. Some common data structures include:

- 1. Arrays: A collection of elements stored at contiguous memory locations, allowing for efficient access to elements using indices.

- 2. Linked Lists: A linear collection of elements where each element points to the next one, allowing for dynamic memory allocation and efficient insertion and deletion operations.

- 3. Stacks: A Last-In-First-Out (LIFO) data structure where elements are inserted and removed from the same end, typically used for function call management, expression evaluation, and undo mechanisms.

- 4. Queues: A First-In-First-Out (FIFO) data structure where elements are inserted at the rear and removed from the front, commonly used in scheduling, buffering, and breadth-first traversal algorithms.

Cont;

- 5. Trees: Hierarchical data structures consisting of nodes connected by edges, with a single root node and branches emanating from it. Examples include binary trees, binary search trees, and AVL trees.

- 6. Graphs: Non-linear data structures consisting of nodes (vertices) connected by edges, representing relationships between objects. Graphs can be directed or undirected and can have cycles or be acyclic.

- 7. Hash Tables: Data structures that implement an associative array abstract data type, using a hash function to map keys to values. Hash tables offer constant-time average case complexity for insertion, deletion, and lookup operations.

- 8. Heaps: Tree-based data structures that satisfy the heap property, such as the binary heap and the Fibonacci heap, commonly used in priority queue implementations and for heap sort.

.

Data Structure

هياكل البيانات هي أساس تنظيم وتخزين البيانات في علوم الكمبيوتر. وهي تحدد طريقة ترتيب البيانات والوصول إليها ومعالجتها في ذاكرة الكمبيوتر. هناك أنواع مختلفة من هياكل البيانات، ولكل منها مزاياها وحالات الاستخدام الخاصة بها. تتضمن بعض هياكل البيانات الشائعة ما يلي:

1. المصفوفات: مجموعة من Arrays العناصر المخزنة في مواقع متجاورة في الذاكرة، مما يسمح بالوصول الفعال إلى العناصر باستخدام الفهارس.

2. القوائم المرتبطة (Linked Lists) : مجموعة خطية من العناصر حيث يشير كل عنصر إلى العنصر التالي، مما يسمح بتخصيص الذاكرة الديناميكية وعمليات الإدراج والحذف الفعالة.

3. الأكوام: بنية بيانات آخر ما يدخل أولاً يخرج A Last-In-First-Out (LIFO) : حيث يتم إدراج العناصر وإزالتها من نفس النهاية، وتستخدم عادةً لإدارة استدعاء الوظائف وتقييم التعبير وآليات التراجع.

4. قوائم الانتظار: بنية بيانات تدخل أولاً تخرج A First-In-First-Out (FIFO) حيث يتم إدراج العناصر في الخلف وإزالتها من الأمام، وتستخدم بشكل شائع في الجدولة، والتخزين المؤقت، وخوارزميات اجتياز العرض أولاً.

الأشجار :Treeهياكل بيانات هرمية تتكون من عقد متصلة بحواف، مع عقدة جذر واحدة وفروع تنبثق عنها. تتضمن الأمثلة الأشجار الثنائية، وأشجار البحث الثنائية.

الرسوم البيانية :Graphsهياكل بيانات غير خطية تتكون من عقد (رؤوس) متصلة بحواف، تمثل العلاقات بين الكائنات. يمكن أن تكون الرسوم البيانية موجهة أو غير موجهة ويمكن أن تحتوي على دورات أو تكون غير دورية.

جداول التجزئة :Hash Tablesجدول التجزئة هو بنية بيانات تُستخدم لتخزين أزواج المفاتيح/القيم. هو أحد بنى المعطيات في علم الحاسوب يملك خصائص المصفوفات الترابطية ((associative array، يستخدم لاسناد قيمة إلى مفتاح ما في ذاكرة الحاسب. والبحث عن قيم محددة بسرعة كبيرة مقارنة ببنى المعطيات الأخرى.

Data from Data Structure Overview

- Data" refers to the information that is organized and manipulated within the structure.
- "Type of data" refers to the specific categories or formats of information being stored or processed. Such as:

1. **Primitive Data Types**: These are basic data types provided by programming languages. Examples include:
    1. Integer (int)
    2. Floating-point number (float)
    3. Character (char)
    4. Boolean (bool)
2. **Derived Data Types**: These are data types that are derived from primitive data types or are composite in nature.

Examples include:

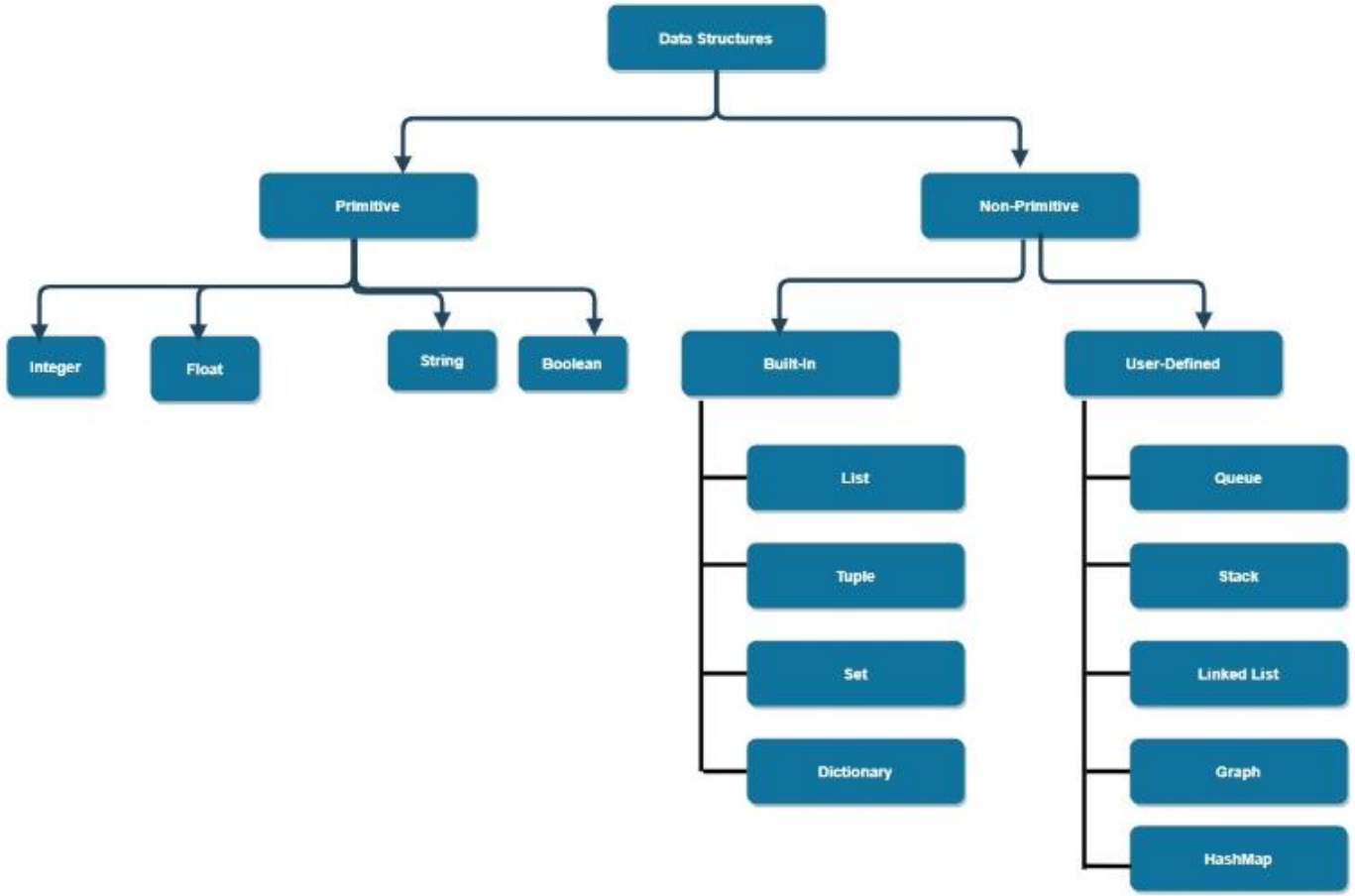
1. Arrays
2. Lists
3. Strings
4. Tuples
5. Sets
6. Dictionaries

1. **User-defined Data Types**: These are data types defined by the user based on their specific requirements.

Examples include:

1. Structures
2. Classes

Type of Data Structure

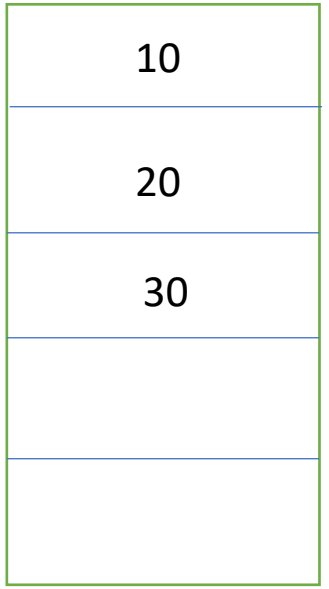


Example

```
# Example of an array in Python
arr = [10, 20, 30, 40, 50]

# Accessing elements by index
print(arr[0])  # Output: 10
print(arr[2])  # Output: 30

# Modifying elements
arr[1] = 25
print(arr)  # Output: [10, 25, 30, 40, 50]

# Iterating through elements
for element in arr:
    print(element)
```

- Example:
- Int A= 10
- Int B=20
- C=30
- D=
- E=

Memory Registry Allocation



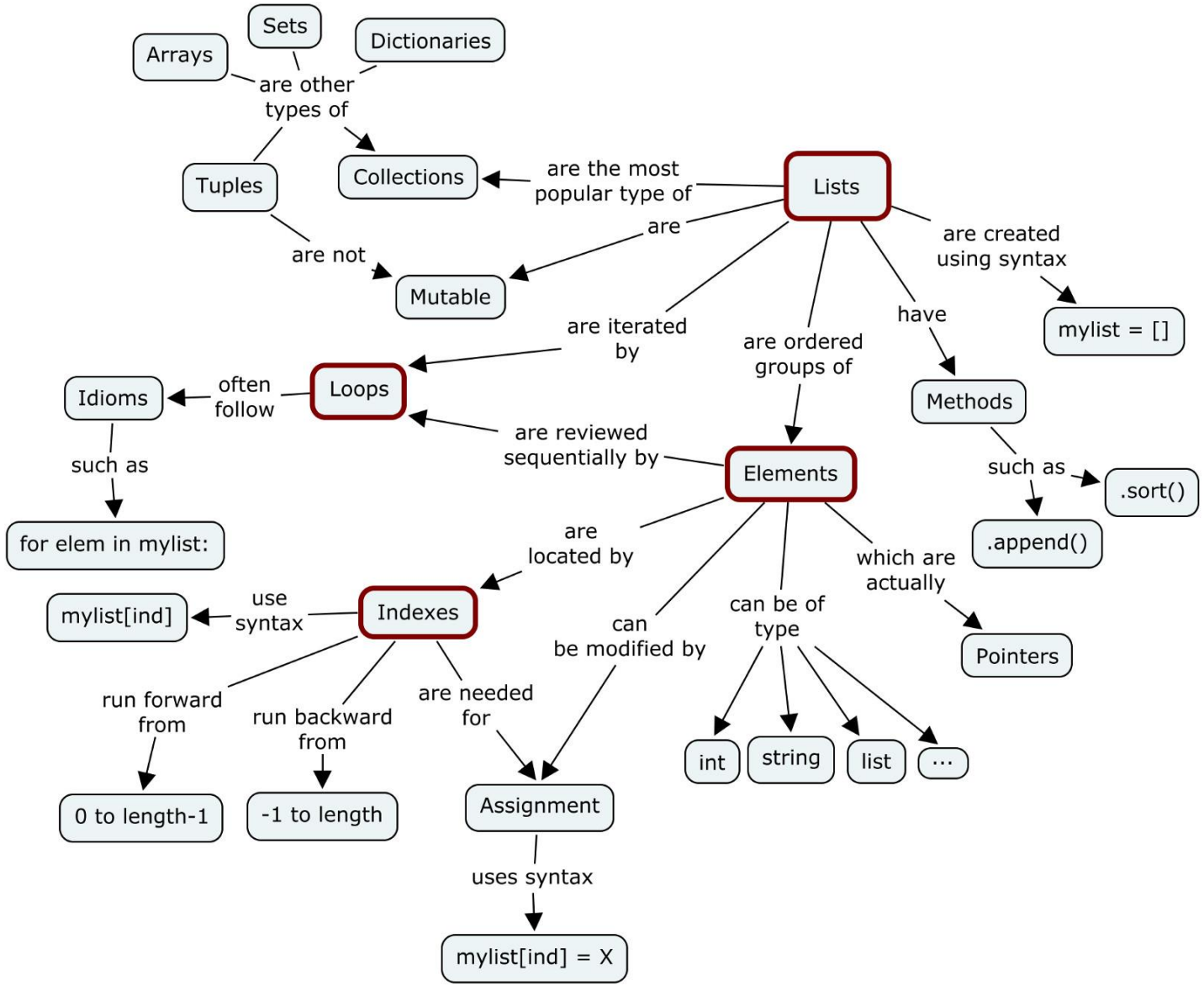
المصفوفات (Arrays)

تحتوي المصفوفة على عناصر داخل كل حاوية وتصطف الحاويات جميعها في تسلسل.

المصفوفة عبارة عن هيكل خطي ثابت الحجم يخزن عناصر متعددة من نفس نوع البيانات بشكل متسلسل.

المصفوفات Arrays : مجموعة من العناصر المخزنة في مواقع متجاورة في الذاكرة، مما يسمح بالوصول الفعال إلى العناصر باستخدام الفهارس.

List



[Python Data Structures (devopedia.org)]

[4737.1513052765.jpg (1600×1310) (devopedia.org)]

Example of creating a list

- Array is a built-in data structure used to store a collection of items.
- Array can contain elements of different data types, including integers, floats, strings, and even other lists.

#```python

```
# Creating a list of integers
my_list = [1, 2, 3, 4, 5]

# Creating a list of strings
names = ["Alice", "Bob", "Charlie", "David"]

# Creating a list with mixed data types
mixed_list = [1, "apple", 3.14, True]

# Creating a list of lists (nested list)
nested_list = [[1, 2], [3, 4], [5, 6]]
```

List

List is ordered and mutable, which means you can change, add, and remove elements after the list is created.
القائمة مرتبة وقابلة للتغيير، مما يعني أنه يمكنك تغيير العناصر وإضافتها وإزالتها بعد إنشاء القائمة.

1) # Modifying elements of a list

my_list[3] = 10

print(my_list) # Output: [1, 2, 3, 10, 5]

2) # Adding elements to a list
names.append("Eve") print(names) # Output:
["Alice", "Bob", "Charlie", "David", "Eve"]

3) # Removing elements from
a list mixed_

list.remove("apple")
print(mixed_list)
# Output: [1, 3.14, True]

4) # Accessing elements of a list

print(my_list[0]) # Output: 1

print(names[2]) # Output: Charlie

Example1

Ex1_list.py - D:\Courses_2024_Data_Strucure\Data Structure\Python_code\Ex1_list.py (3.9.0)

File Edit Format Run Options Window Help

```
# Data Strucure Course (2024)
# Example of an array
List = [10, 20, 30, 40, 50]



# Accessing elements by index
print(List[0])  # Output: 10
print(List[2])  # Output: 30

# Modifying elements
List[1] = 25
print(List)  # Output: [10, 25, 30, 40, 50]

# Iterating through elements
for element in List:
    print(element)
```

```
# Example of an array in Python
arr = [10, 20, 30, 40, 50]

# Accessing elements by index
print(arr[0])  # Output: 10
print(arr[2])  # Output: 30

# Modifying elements
arr[1] = 25
print(arr)  # Output: [10, 25, 30, 40, 50]

# Iterating through elements
for element in arr:
    print(element)
```

Example

- >>> # Example of usPython 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
- Type "help", "copyright", "credits" or "license()" for more information.
- ing Python array to store a list of fruits
- fruits = ['Apple', 'Banana', 'Orange', 'Grapes', 'Watermelon']
- # Accessing elements by index
- print("First fruit:", fruits[0]) # Output: First fruit: Apple
- print("Third fruit:", fruits[2])   # Output: Third fruit: Orange
- # Modifying elements
- fruits[1] = 'Mango'
- print("Modified fruits list:", fruits) # Output: Modified fruits list: ['Apple', 'Mango', 'Orange', 'Grapes', 'Watermelon']
- # Iterating through elements
- print("Fruits list:")
- for fruit in fruits:
- print(fruit)

Exampe 3 (Boolean data type)

```
pass_fail = []
Avg = [90,80,50,30,85,70,69]
pass_threshold  = 50
 for score in Avg:
   if score  >= pass_threshold:
     pass_fail.append(True)
   else:
     pass_fail.append(False)
 for i, status in enumerate(pass_fail):
    if status:
      print(f"Students {i+1}: Pass")
    else:
      print(f"students {i+1}: Fail")
```

```
pass_fail = []
Avg = [90,80,50,30,85,70,69]
pass_threshold  = 50
for score in Avg:
    if score  >= pass_threshold:
       pass_fail.append(True)
    else:
        pass_fail.append(False)

for i, status in enumerate(pass_fail):
    if status:
        print(f"Students {i+1}: Pass")
    else:
        print(f"students {i+1}: Fail")
```

```
Students 1: Pass
Students 2: Pass
students 3: Fail
students 4: Fail
Students 5: Pass
Students 6: Pass
Students 7: Pass
```

```
# Define the array to store Boolean values for pass or fail
pass_fail = []
# Suppose we have 5 students and their average scores
average_scores = [85, 62, 75, 90, 55]
# Define the passing threshold
passing_threshold = 70
# Iterate through each student's average score
for score in average_scores:
    # Check if the score is above the passing threshold
    if score >= passing_threshold:
        pass_fail.append(True)  # True indicates pass
    else:
        pass_fail.append(False)  # False indicates fail
# Print the pass/fail status of each student
for i, status in enumerate(pass_fail):
    if status:
        print(f"Student {i+1}: Pass")
    else:
        print(f"Student {i+1}: Fail")
```

إضافة عناصر Append() إلى القائمة باستخدام

- دالة ``append()`` في Pythonتُستخدم لإضافة عنصر إلى نهاية قائمة محددة.
- Example

- 1-تعريف قائمة فارغة
- my_list =[]

- #إضافة عناصر إلى القائمة باستخدام append()
- my_list.append(1)
- my_list.append(2)
- my_list.append(3)
- Print(my_list)

- في هذا المثال، تم إنشاء قائمة فارغة ``my_list``، ثم تم استخدام دالة ``append()`` لإضافة الأعداد 1، 2، و 3 إلى نهاية القائمة. سيتم طباعة القائمة بعد الإضافة وستحتوي على العناصر المضافة.

Assignment

- The result needed to be implemented as:
- Student 1: Score - 90, Result - Pass
- Student 2: Score - 80, Result - Pass
- Student 3: Score - 50, Result - Fail
- Student 4: Score - 30, Result - Fail
- Student 5: Score - 85, Result - Pass
- Student 6: Score - 70, Result - Pass
- Student 7: Score - 69, Result - Pass