**#COURSE NAME**

**#Class name**

**"#PRACTICAL\_NUMBER"**

***BY***

**"aseem" (#Registration Number)**

**SUBMITTED TO**

**#teacher name**

****

**SCHOOL OF SCIENCES**

**2025-2026**

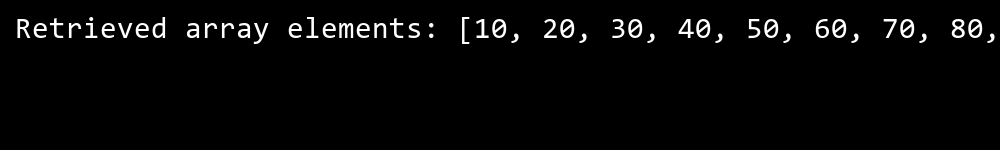
## **QUESTION 1**

1. Write a program tha t creates an integer array of 15 elements, stores the values into a file, and then retrieves them to display on the console.

### **Code Solution**

import array  
import pickle  
  
numbers = array.array('i', [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150])  
  
with open('array\_data.bin', 'wb') as file:  
 pickle.dump(numbers, file)  
  
with open('array\_data.bin', 'rb') as file:  
 loaded\_numbers = pickle.load(file)  
  
print("Retrieved array elements:", list(loaded\_numbers))

### **FINAL Output**



## **QUESTION 2**

2. Write a program to input two integers and divide them. Use a try -catch block to handle the DivideByZeroException and display an appropriate message. Further, if the data type of the elements do not match with defined type then throw an exception too.

### **Code Solution**

try:  
 num1 = 6  
 num2 = 0  
 if not isinstance(num1, int) or not isinstance(num2, int):  
 raise TypeError("Both numbers must be integers")  
 result = num1 / num2  
 print(f"Result of division: {result}")  
except ZeroDivisionError:  
 print("Error: Cannot divide by zero")  
except TypeError as te:  
 print(f"Error: {te}")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### **FINAL Output**



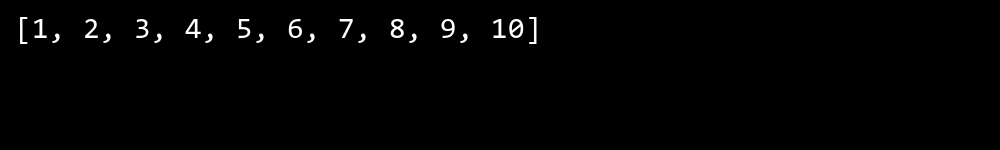
## **QUESTION 3**

3. Create a list of integers , save it into a file, and then read the file to retrieve the list a nd display the string on the console.

### **Code Solution**

my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
with open('numbers.txt', 'w') as file:  
 file.write(str(my\_list))  
with open('numbers.txt', 'r') as file:  
 loaded\_list = eval(file.read())  
print(loaded\_list)

### **FINAL Output**



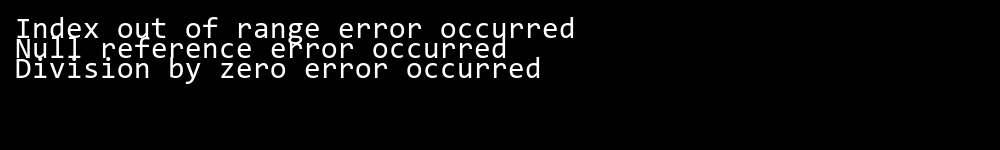
## **QUESTION 4**

4. Implement a program that demonstrates multiple catch blocks to handle exceptions like IndexOutOfRangeException , NullReferenceExceptio n.

### **Code Solution**

try:  
 numbers = [1, 2, 3]  
 print(numbers[5])  
except IndexError:  
 print("Index out of range error occurred")  
  
try:  
 my\_list = None  
 print(my\_list[0])  
except TypeError:  
 print("Null reference error occurred")  
  
try:  
 result = 10/0  
except ZeroDivisionError:  
 print("Division by zero error occurred")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### **FINAL Output**



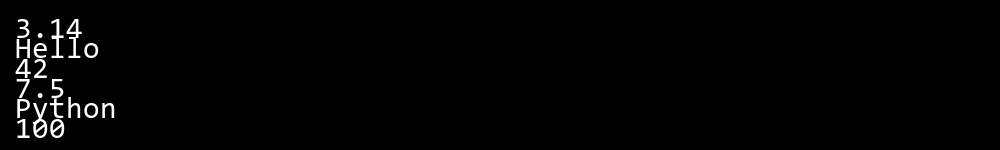
## **QUESTION 5**

5. Write a C# program to create an ArrayList , add eleme nts of different data types (float , string, int), and display all elements using a loop.

### **Code Solution**

my\_list = []  
my\_list.append(3.14)  
my\_list.append("Hello")  
my\_list.append(42)  
my\_list.append(7.5)  
my\_list.append("Python")  
my\_list.append(100)  
  
for item in my\_list:  
 print(item)

### **FINAL Output**



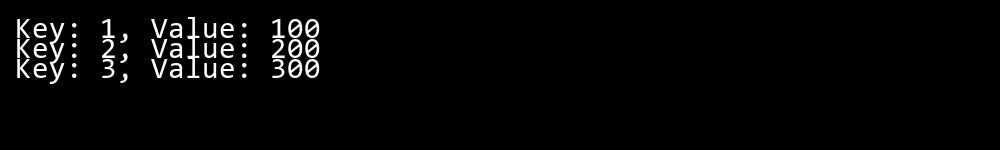
## **QUESTION 6**

6. Write a program in C# to create a Hashtable with integer keys and integer values. Insert three key -value pairs and display them using a loop.

### **Code Solution**

hashtable = {}  
hashtable[1] = 100  
hashtable[2] = 200  
hashtable[3] = 300  
for key, value in hashtable.items():  
 print(f"Key: {key}, Value: {value}")

### **FINAL Output**



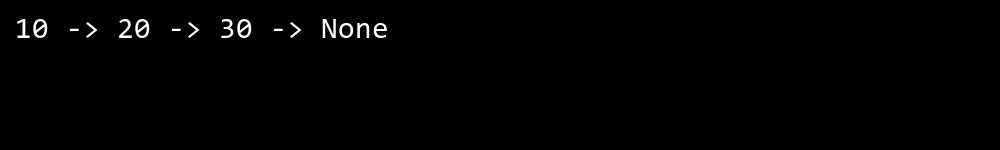
## **QUESTION 7**

7. Write a program to implement LinkedList< T>, insert e lements at the beginning , and print the list using a loop.

### **Code Solution**

class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class LinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
   
 def insert\_at\_beginning(self, data):  
 new\_node = Node(data)  
 new\_node.next = self.head  
 self.head = new\_node  
   
 def print\_list(self):  
 current = self.head  
 while current:  
 print(current.data, end=" -> ")  
 current = current.next  
 print("None")  
  
linked\_list = LinkedList()  
linked\_list.insert\_at\_beginning(30)  
linked\_list.insert\_at\_beginning(20)  
linked\_list.insert\_at\_beginning(10)  
linked\_list.print\_list()

### **FINAL Output**



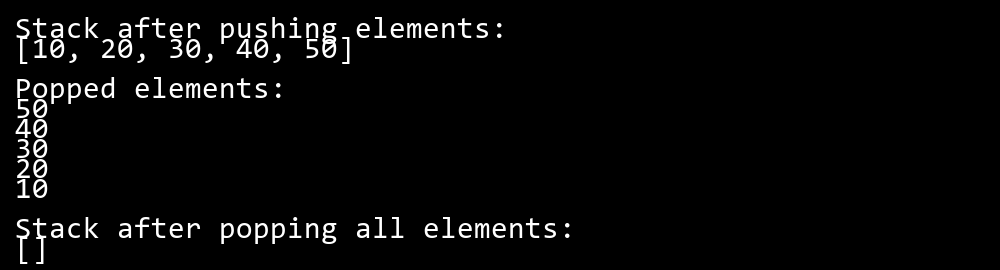
## **QUESTION 8**

8. Write a program to implement Stack , insert five elements and remove them .

### **Code Solution**

class Stack:  
 def \_\_init\_\_(self):  
 self.stack = []  
   
 def push(self, item):  
 self.stack.append(item)  
   
 def pop(self):  
 if not self.is\_empty():  
 return self.stack.pop()  
   
 def is\_empty(self):  
 return len(self.stack) == 0  
   
 def peek(self):  
 if not self.is\_empty():  
 return self.stack[-1]  
   
 def size(self):  
 return len(self.stack)  
   
 def display(self):  
 print(self.stack)  
  
stack = Stack()  
  
stack.push(10)  
stack.push(20)  
stack.push(30)  
stack.push(40)  
stack.push(50)  
  
print("Stack after pushing elements:")  
stack.display()  
  
print("\nPopped elements:")  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())  
  
print("\nStack after popping all elements:")  
stack.display()

### **FINAL Output**



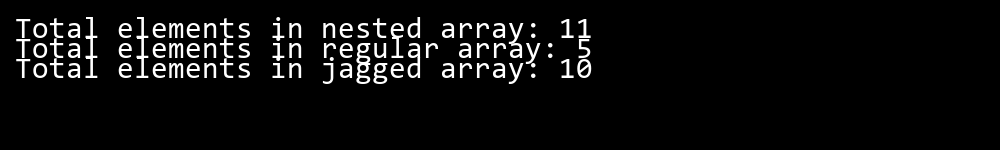
## **QUESTION 9**

9. Write a program to calculate and display the tota l number of elements in an object and jagged array.

### **Code Solution**

array = [1, [2, 3], [4, [5, 6]], 7, [8, 9, [10, 11]]]  
  
def count\_elements(arr):  
 total = 0  
 for element in arr:  
 if isinstance(element, list):  
 total += count\_elements(element)  
 else:  
 total += 1  
 return total  
  
regular\_array = [1, 2, 3, 4, 5]  
jagged\_array = [[1, 2], [3, 4, 5], [6], [7, 8, 9, 10]]  
  
print(f"Total elements in nested array: {count\_elements(array)}")  
print(f"Total elements in regular array: {len(regular\_array)}")  
print(f"Total elements in jagged array: {sum(len(row) for row in jagged\_array)}")

### **FINAL Output**



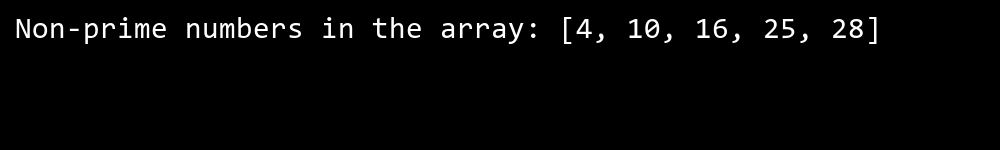
## **QUESTION 10**

10. Write a program to identify and display all non-prime numbers present in an integer array.

### **Code Solution**

def check\_prime(n):  
 if n < 2:  
 return True  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return True  
 return False  
  
array = [4, 7, 10, 13, 16, 19, 23, 25, 28, 31]  
non\_prime = []  
  
for num in array:  
 if check\_prime(num):  
 non\_prime.append(num)  
  
print("Non-prime numbers in the array:", non\_prime)

### **FINAL Output**



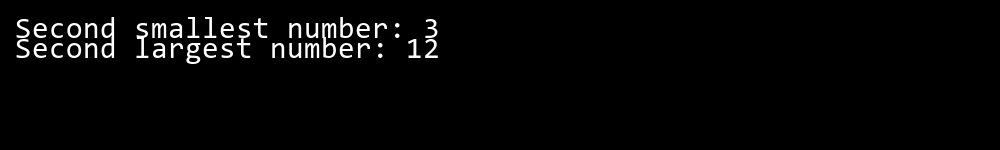
## **QUESTION 11**

11. Write a program to find and displ ay the second largest and smallest numbers in an array.

### **Code Solution**

def find\_second\_extremes(arr):  
 if len(arr) < 2:  
 return None, None  
 sorted\_arr = sorted(arr)  
 second\_smallest = sorted\_arr[1]  
 second\_largest = sorted\_arr[-2]  
 return second\_smallest, second\_largest  
  
numbers = [10, 5, 8, 12, 3, 7, 9, 15, 1, 6]  
second\_min, second\_max = find\_second\_extremes(numbers)  
print(f"Second smallest number: {second\_min}")  
print(f"Second largest number: {second\_max}")

### **FINAL Output**



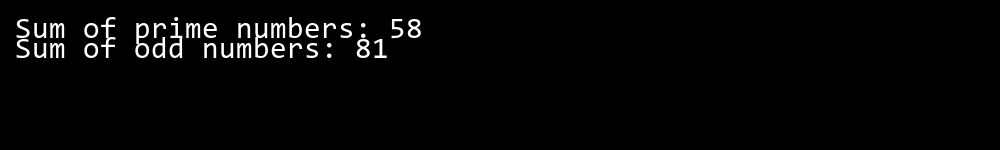
## **QUESTION 12**

12. Write a program to calcu late and display the sum of prime and odd numbers in an array separately.

### **Code Solution**

def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
def is\_odd(n):  
 return n % 2 != 0  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17]  
  
prime\_sum = sum(num for num in array if is\_prime(num))  
odd\_sum = sum(num for num in array if is\_odd(num))  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### **FINAL Output**



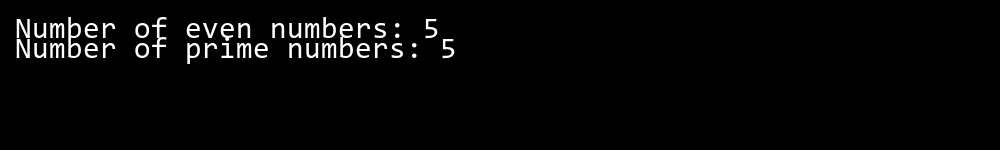
## **QUESTION 13**

13. Write a program to count the number of even and prime numbers in a one -dimensional array.

### **Code Solution**

def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
def count\_even\_and\_prime(arr):  
 even\_count = 0  
 prime\_count = 0  
   
 for num in arr:  
 if num % 2 == 0:  
 even\_count += 1  
 if is\_prime(num):  
 prime\_count += 1  
   
 return even\_count, prime\_count  
  
array = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]  
even\_numbers, prime\_numbers = count\_even\_and\_prime(array)  
  
print(f"Number of even numbers: {even\_numbers}")  
print(f"Number of prime numbers: {prime\_numbers}")

### **FINAL Output**



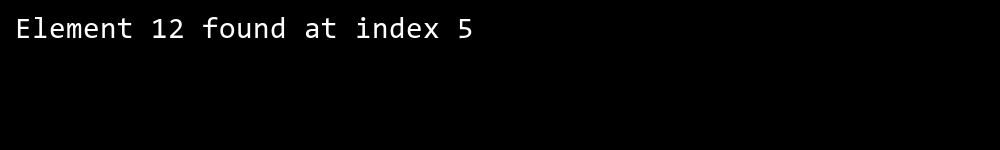
## **QUESTION 14**

14. Implement a program to search for a specific element in an array using binary search .

### **Code Solution**

def binary\_search(arr, target):  
 left = 0  
 right = len(arr) - 1  
   
 while left <= right:  
 mid = (left + right) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
array = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]  
target = 12  
  
result = binary\_search(array, target)  
  
if result != -1:  
 print(f"Element {target} found at index {result}")  
else:  
 print(f"Element {target} not found in the array")

### **FINAL Output**



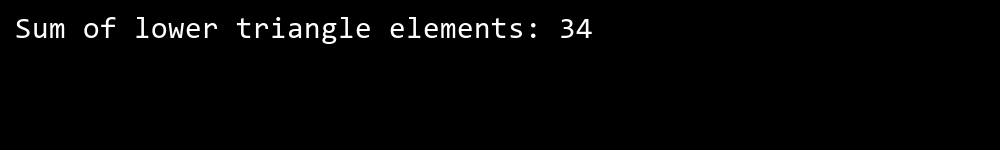
## **QUESTION 15**

15. Write a program to calculate the sum of the lower triangle elements of a square matrix.

### **Code Solution**

def sum\_lower\_triangle(matrix):  
 n = len(matrix)  
 total = 0  
 for i in range(n):  
 for j in range(i + 1):  
 total += matrix[i][j]  
 return total  
  
matrix = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]  
]  
  
result = sum\_lower\_triangle(matrix)  
print("Sum of lower triangle elements:", result)

### **FINAL Output**



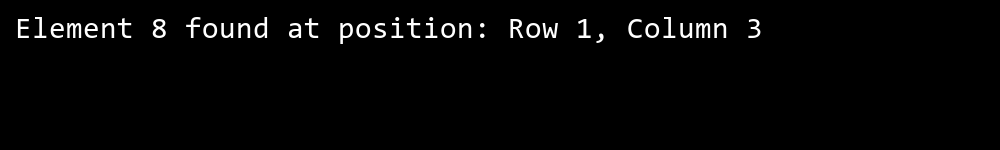
## **QUESTION 16**

16. Write a C# program to perform linear search on a sorted jagged array .

### **Code Solution**

def linear\_search\_jagged(jagged\_array, target):  
 for i in range(len(jagged\_array)):  
 for j in range(len(jagged\_array[i])):  
 if jagged\_array[i][j] == target:  
 return (i, j)  
 return (-1, -1)  
  
jagged\_array = [  
 [1, 3, 5],  
 [2, 4, 6, 8],  
 [7, 9],  
 [10, 11, 12, 13, 14]  
]  
  
target = 8  
result = linear\_search\_jagged(jagged\_array, target)  
  
if result != (-1, -1):  
 print(f"Element {target} found at position: Row {result[0]}, Column {result[1]}")  
else:  
 print(f"Element {target} not found in the array")

### **FINAL Output**



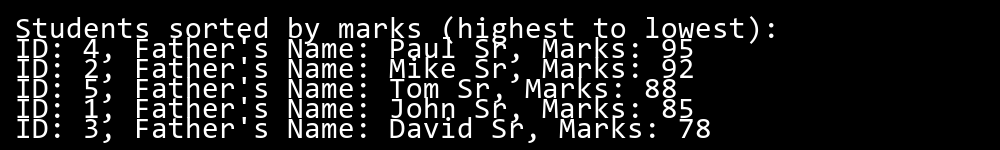
## **QUESTION 17**

17. Create a Student class with properties (ID, Father\_ Name, Marks). Store multiple students in an object array and sort them by Marks. Further, s tore sorted students in a Linked List<T> and display them.

### **Code Solution**

class Student:  
 def \_\_init\_\_(self, ID, Father\_Name, Marks):  
 self.ID = ID  
 self.Father\_Name = Father\_Name  
 self.Marks = Marks  
  
class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class LinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
  
 def append(self, data):  
 new\_node = Node(data)  
 if not self.head:  
 self.head = new\_node  
 return  
 current = self.head  
 while current.next:  
 current = current.next  
 current.next = new\_node  
  
 def display(self):  
 current = self.head  
 while current:  
 print(f"ID: {current.data.ID}, Father's Name: {current.data.Father\_Name}, Marks: {current.data.Marks}")  
 current = current.next  
  
students = [  
 Student(1, "John Sr", 85),  
 Student(2, "Mike Sr", 92),  
 Student(3, "David Sr", 78),  
 Student(4, "Paul Sr", 95),  
 Student(5, "Tom Sr", 88)  
]  
  
sorted\_students = sorted(students, key=lambda x: x.Marks, reverse=True)  
  
linked\_list = LinkedList()  
for student in sorted\_students:  
 linked\_list.append(student)  
  
print("Students sorted by marks (highest to lowest):")  
linked\_list.display()

### **FINAL Output**



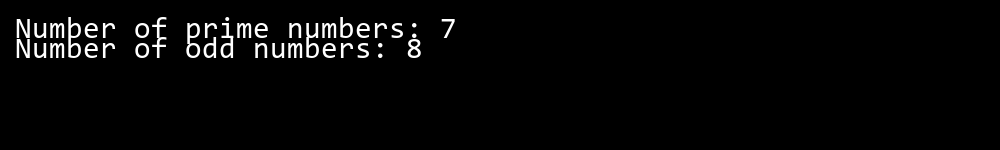
## **QUESTION 18**

18. Write a program to count the number of prime and odd numbers in a one -dimensional array.

### **Code Solution**

def is\_prime(n):  
 if n < 2:  
 return False  
 for i in range(2, int(n \*\* 0.5) + 1):  
 if n % i == 0:  
 return False  
 return True  
  
def count\_prime\_odd(arr):  
 prime\_count = 0  
 odd\_count = 0  
 for num in arr:  
 if is\_prime(num):  
 prime\_count += 1  
 if num % 2 != 0:  
 odd\_count += 1  
 return prime\_count, odd\_count  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 17]  
prime\_numbers, odd\_numbers = count\_prime\_odd(array)  
print(f"Number of prime numbers: {prime\_numbers}")  
print(f"Number of odd numbers: {odd\_numbers}")

### **FINAL Output**



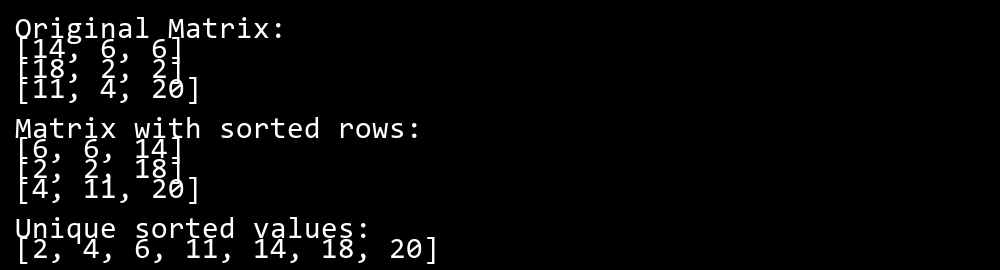
## **QUESTION 19**

19. Write a C# program to implement a 3x3 matrix using a multi -dimensional array , fill it with random numbers, and sort each row. Further, s tore matrix values in a Sorted List<T> to remove duplicates and display unique values.

### **Code Solution**

import random  
  
matrix = [[0 for \_ in range(3)] for \_ in range(3)]  
  
for i in range(3):  
 for j in range(3):  
 matrix[i][j] = random.randint(1, 20)  
  
print("Original Matrix:")  
for row in matrix:  
 print(row)  
  
for i in range(3):  
 matrix[i].sort()  
  
print("\nMatrix with sorted rows:")  
for row in matrix:  
 print(row)  
  
unique\_values = sorted(list(set([num for row in matrix for num in row])))  
  
print("\nUnique sorted values:")  
print(unique\_values)

### **FINAL Output**



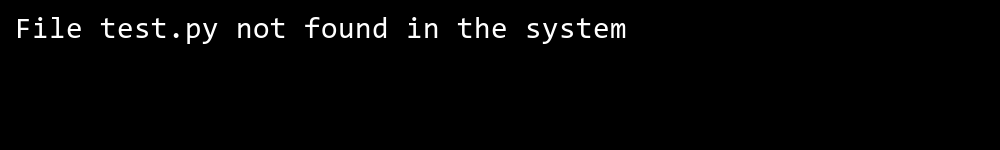
## **QUESTION 20**

20. Write a C# program to implement a program that reads an array of filenames and searches for a specific file in the system. Further, s tore valid file names in a Directory collection and allow the user to retrieve details about a specific file.

### **Code Solution**

import os  
from collections import defaultdict  
  
file\_list = ['example.txt', 'test.py', 'data.csv', 'document.docx', 'image.jpg']  
search\_file = 'test.py'  
file\_directory = defaultdict(dict)  
  
for filename in file\_list:  
 if os.path.exists(filename):  
 file\_stats = os.stat(filename)  
 file\_directory[filename] = {  
 'size': file\_stats.st\_size,  
 'created': file\_stats.st\_ctime,  
 'modified': file\_stats.st\_mtime,  
 'path': os.path.abspath(filename)  
 }  
  
if search\_file in file\_directory:  
 file\_details = file\_directory[search\_file]  
 print(f"File found: {search\_file}")  
 print(f"Size: {file\_details['size']} bytes")  
 print(f"Created: {file\_details['created']}")  
 print(f"Modified: {file\_details['modified']}")  
 print(f"Path: {file\_details['path']}")  
else:  
 print(f"File {search\_file} not found in the system")

### **FINAL Output**



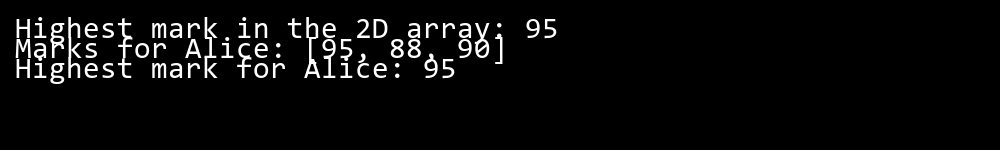
## **QUESTION 21**

21. Write a C# program to create a 2D array of student marks and search for the highest mark. Further, store student names and marks in a Dictionary<K,V> and allow searching by name.

### **Code Solution**

student\_marks = [[85, 92, 78], [95, 88, 90], [76, 85, 92], [89, 94, 87]]  
  
highest\_mark = float('-inf')  
for row in student\_marks:  
 current\_max = max(row)  
 if current\_max > highest\_mark:  
 highest\_mark = current\_max  
  
student\_data = {  
 "John": [85, 92, 78],  
 "Alice": [95, 88, 90],  
 "Bob": [76, 85, 92],  
 "Emma": [89, 94, 87]  
}  
  
print(f"Highest mark in the 2D array: {highest\_mark}")  
  
search\_name = "Alice"  
if search\_name in student\_data:  
 student\_marks = student\_data[search\_name]  
 print(f"Marks for {search\_name}: {student\_marks}")  
 print(f"Highest mark for {search\_name}: {max(student\_marks)}")  
else:  
 print(f"Student {search\_name} not found")

### **FINAL Output**



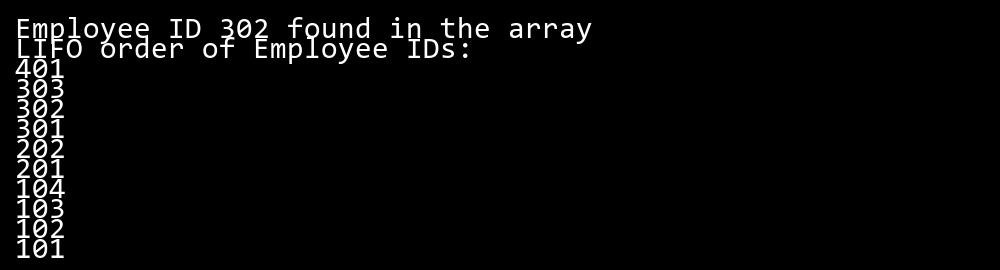
## **QUESTION 22**

22. Write a C# program to implement Binary Search in a jagged array of employee IDs. Further, s tore IDs in a Stack<T> , push/pop operations for LIFO retrieval.

### **Code Solution**

def binary\_search(arr, target):  
 left = 0  
 right = len(arr) - 1  
 while left <= right:  
 mid = (left + right) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
class Stack:  
 def \_\_init\_\_(self):  
 self.items = []  
   
 def push(self, item):  
 self.items.append(item)  
   
 def pop(self):  
 if not self.is\_empty():  
 return self.items.pop()  
 return None  
   
 def is\_empty(self):  
 return len(self.items) == 0  
   
 def peek(self):  
 if not self.is\_empty():  
 return self.items[-1]  
 return None  
  
jagged\_array = [  
 [101, 102, 103, 104],  
 [201, 202],  
 [301, 302, 303],  
 [401]  
]  
  
stack = Stack()  
  
for row in jagged\_array:  
 row.sort()  
 for emp\_id in row:  
 stack.push(emp\_id)  
  
search\_id = 302  
found = False  
  
temp\_stack = Stack()  
while not stack.is\_empty():  
 current\_id = stack.pop()  
 temp\_stack.push(current\_id)  
 if current\_id == search\_id:  
 found = True  
 break  
  
print(f"Employee ID {search\_id} {'found' if found else 'not found'} in the array")  
  
while not temp\_stack.is\_empty():  
 stack.push(temp\_stack.pop())  
  
print("LIFO order of Employee IDs:")  
while not stack.is\_empty():  
 print(stack.pop())

### **FINAL Output**



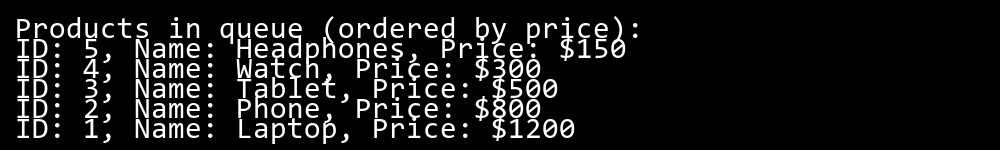
## **QUESTION 23**

23. Write a C# program to create a Product class (ID, Name, Price) and store o bjects in an array by price. Further, u se a Queue<T> to manage product processing (FIFO order).

### **Code Solution**

class Product:  
 def \_\_init\_\_(self, id, name, price):  
 self.id = id  
 self.name = name  
 self.price = price  
  
class Queue:  
 def \_\_init\_\_(self):  
 self.items = []  
  
 def enqueue(self, item):  
 self.items.append(item)  
  
 def dequeue(self):  
 if not self.is\_empty():  
 return self.items.pop(0)  
 return None  
  
 def is\_empty(self):  
 return len(self.items) == 0  
  
 def size(self):  
 return len(self.items)  
  
products = [  
 Product(1, "Laptop", 1200),  
 Product(2, "Phone", 800),  
 Product(3, "Tablet", 500),  
 Product(4, "Watch", 300),  
 Product(5, "Headphones", 150)  
]  
  
sorted\_products = sorted(products, key=lambda x: x.price)  
  
product\_queue = Queue()  
for product in sorted\_products:  
 product\_queue.enqueue(product)  
  
print("Products in queue (ordered by price):")  
while not product\_queue.is\_empty():  
 product = product\_queue.dequeue()  
 print(f"ID: {product.id}, Name: {product.name}, Price: ${product.price}")

### **FINAL Output**



## **QUESTION 24**

24. Write a program to calculate the sum of the diagonal elements of a square matrix.

### **Code Solution**

matrix = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]  
]  
diagonal\_sum = 0  
for i in range(len(matrix)):  
 diagonal\_sum += matrix[i][i]  
print(diagonal\_sum)

### **FINAL Output**

