**SOFTWARE ENGENEERING**

**3 bca b**

**"Practical - 4"**

***BY***

**"Tharan" (23215134)**

**SUBMITTED TO**

**prateek singh**

****

**SCHOOL OF SCIENCES**

**2025-2026**

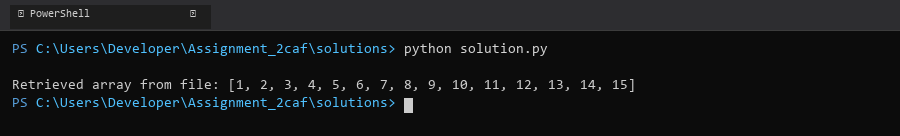
## ***QUESTION 1***

***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 numpy as np  
  
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])  
  
with open('array\_data.txt', 'w') as file:  
 for number in array:  
 file.write(str(number) + '\n')  
  
stored\_array = []  
with open('array\_data.txt', 'r') as file:  
 for line in file:  
 stored\_array.append(int(line.strip()))  
  
print("Retrieved array from file:", stored\_array)

### ***FINAL Output***



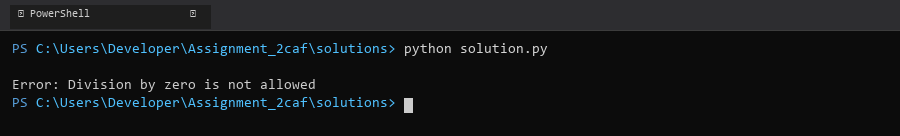
## ***QUESTION 2***

***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 = 10  
 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: Division by zero is not allowed")  
except TypeError as e:  
 print(f"Error: {e}")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### ***FINAL Output***



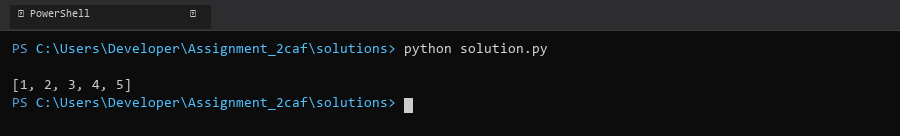
## ***QUESTION 3***

***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***

original\_list = [1, 2, 3, 4, 5]  
  
with open('numbers.txt', 'w') as file:  
 for number in original\_list:  
 file.write(str(number) + '\n')  
  
retrieved\_list = []  
with open('numbers.txt', 'r') as file:  
 for line in file:  
 retrieved\_list.append(int(line.strip()))  
  
print(retrieved\_list)

### ***FINAL Output***



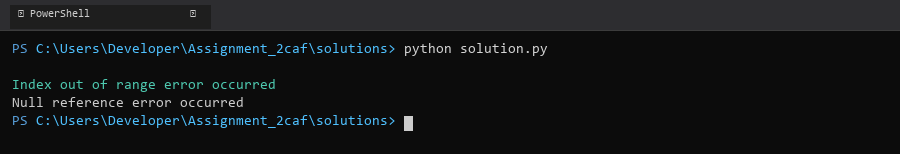
## ***QUESTION 4***

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

### ***Code Solution***

def divide\_numbers():  
 numbers = [10, 20, 0, 40, 50]  
 string\_value = None  
   
 try:  
 result = numbers[10] / numbers[2]  
 print(result)  
 except IndexError:  
 print("Index out of range error occurred")  
 except ZeroDivisionError:  
 print("Division by zero error occurred")  
 except TypeError:  
 print("Type error occurred")  
 except AttributeError:  
 print("Attribute error occurred")  
 except Exception:  
 print("An unknown error occurred")  
  
def access\_null\_reference():  
 try:  
 string\_value = None  
 length = len(string\_value)  
 print(length)  
 except TypeError:  
 print("Null reference error occurred")  
  
divide\_numbers()  
access\_null\_reference()

### ***FINAL Output***



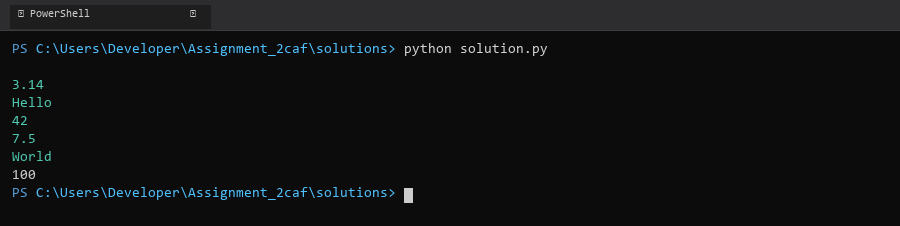
## ***QUESTION 5***

***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***

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

### ***FINAL Output***



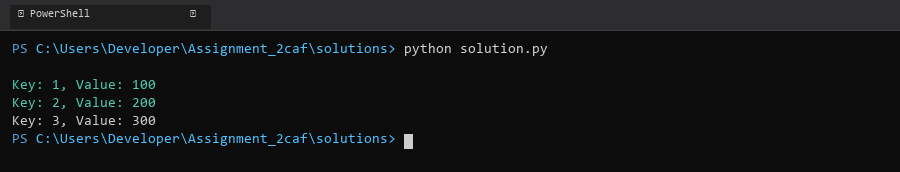
## ***QUESTION 6***

***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***

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

### ***FINAL Output***



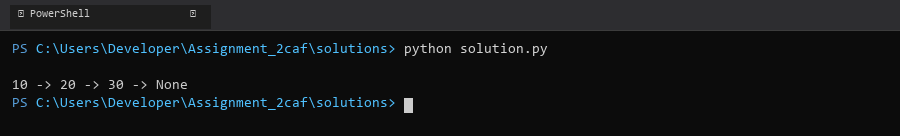
## ***QUESTION 7***

***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***



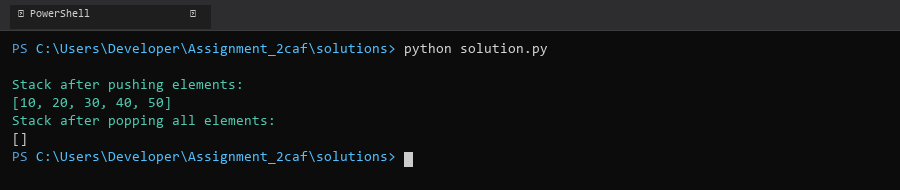
## ***QUESTION 8***

***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)  
  
s = Stack()  
s.push(10)  
s.push(20)  
s.push(30)  
s.push(40)  
s.push(50)  
  
print("Stack after pushing elements:")  
s.display()  
  
s.pop()  
s.pop()  
s.pop()  
s.pop()  
s.pop()  
  
print("Stack after popping all elements:")  
s.display()

### ***FINAL Output***



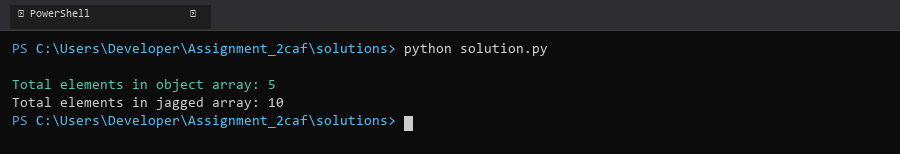
## ***QUESTION 9***

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

### ***Code Solution***

object\_array = [1, 2, 3, 4, 5]  
jagged\_array = [[1, 2, 3], [4, 5], [6, 7, 8, 9], [10]]  
  
object\_count = len(object\_array)  
  
jagged\_count = sum(len(row) for row in jagged\_array)  
  
print("Total elements in object array:", object\_count)  
print("Total elements in jagged array:", jagged\_count)

### ***FINAL Output***



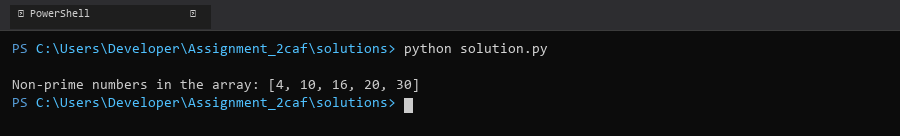
## ***QUESTION 10***

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

### ***Code Solution***

def is\_not\_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  
  
arr = [4, 7, 10, 13, 16, 19, 20, 23, 29, 30]  
  
non\_prime\_numbers = []  
for num in arr:  
 if is\_not\_prime(num):  
 non\_prime\_numbers.append(num)  
  
print("Non-prime numbers in the array:", non\_prime\_numbers)

### ***FINAL Output***



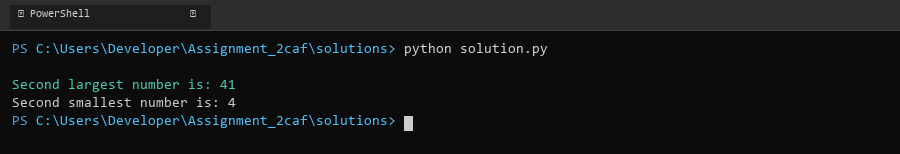
## ***QUESTION 11***

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

### ***Code Solution***

def second\_largest\_smallest(arr):  
 if len(arr) < 2:  
 return None, None  
   
 sorted\_arr = sorted(arr)  
 second\_smallest = sorted\_arr[1]  
 second\_largest = sorted\_arr[-2]  
   
 return second\_largest, second\_smallest  
  
array = [12, 45, 2, 41, 31, 10, 8, 6, 4]  
second\_max, second\_min = second\_largest\_smallest(array)  
print(f"Second largest number is: {second\_max}")  
print(f"Second smallest number is: {second\_min}")

### ***FINAL Output***



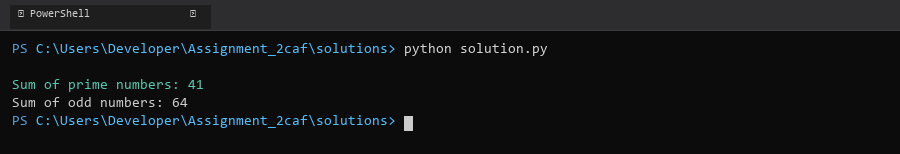
## ***QUESTION 12***

***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  
  
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]  
  
prime\_sum = 0  
odd\_sum = 0  
  
for num in numbers:  
 if is\_prime(num):  
 prime\_sum += num  
 if is\_odd(num):  
 odd\_sum += num  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### ***FINAL Output***



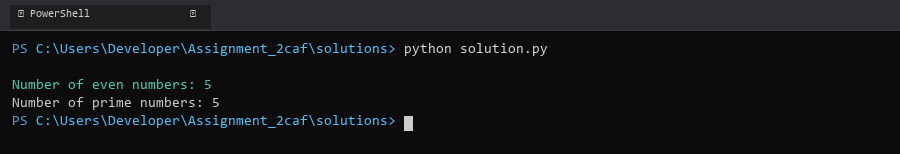
## ***QUESTION 13***

***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\_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\_prime(array)  
  
print(f"Number of even numbers: {even\_numbers}")  
print(f"Number of prime numbers: {prime\_numbers}")

### ***FINAL Output***



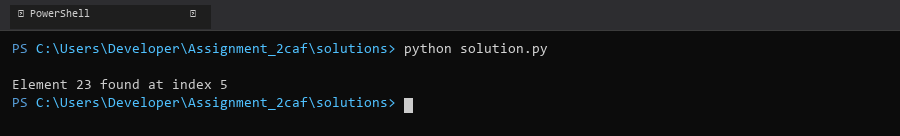
## ***QUESTION 14***

***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, 5, 8, 12, 16, 23, 38, 45, 56, 72]  
target = 23  
  
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***



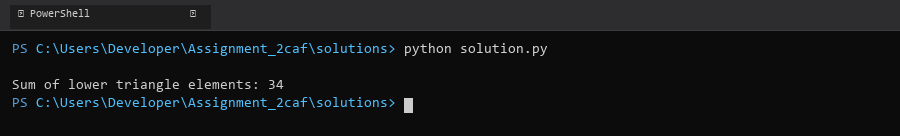
## ***QUESTION 15***

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

### ***Code Solution***

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

### ***FINAL Output***



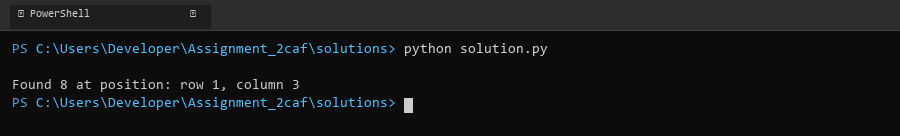
## ***QUESTION 16***

***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 True, i, j  
 return False, -1, -1  
  
jagged\_array = [  
 [1, 3, 5],  
 [2, 4, 6, 8],  
 [7, 9],  
 [10, 11, 12, 13, 14]  
]  
  
target = 8  
found, row, col = linear\_search\_jagged(jagged\_array, target)  
  
if found:  
 print(f"Found {target} at position: row {row}, column {col}")  
else:  
 print(f"{target} not found in the array")

### ***FINAL Output***



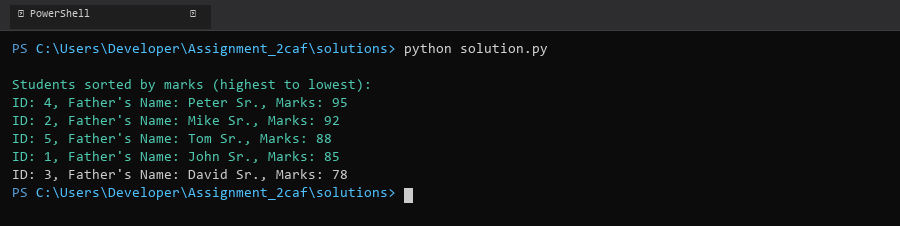
## ***QUESTION 17***

***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=None):  
 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, "Peter 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***



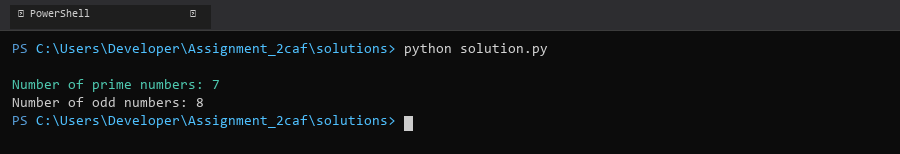
## ***QUESTION 18***

***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 is\_odd(n):  
 return n % 2 != 0  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 17]  
  
prime\_count = sum(1 for num in array if is\_prime(num))  
odd\_count = sum(1 for num in array if is\_odd(num))  
  
print(f"Number of prime numbers: {prime\_count}")  
print(f"Number of odd numbers: {odd\_count}")

### ***FINAL Output***



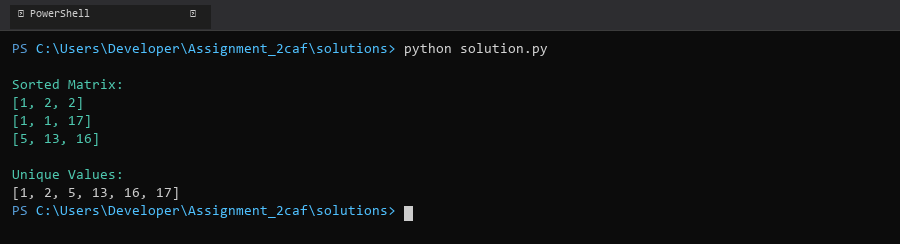
## ***QUESTION 19***

***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 = [[random.randint(1, 20) for \_ in range(3)] for \_ in range(3)]  
  
for row in matrix:  
 row.sort()  
  
print("Sorted Matrix:")  
for row in matrix:  
 print(row)  
  
unique\_values = sorted(list(set([num for row in matrix for num in row])))  
  
print("\nUnique Values:")  
print(unique\_values)

### ***FINAL Output***



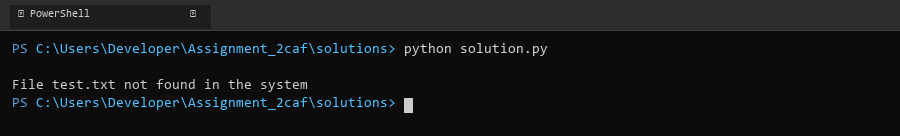
## ***QUESTION 20***

***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 pathlib import Path  
  
filenames = ['test.txt', 'document.pdf', 'image.jpg', 'script.py', 'data.csv', 'notes.txt', 'report.docx']  
search\_file = 'test.txt'  
directory = {}  
  
for filename in filenames:  
 file\_path = Path(filename)  
 if file\_path.exists():  
 file\_stats = os.stat(filename)  
 directory[filename] = {  
 'size': file\_stats.st\_size,  
 'created': file\_stats.st\_ctime,  
 'modified': file\_stats.st\_mtime,  
 'path': str(file\_path.absolute())  
 }  
  
specific\_file = filenames[7 % len(filenames)]  
if specific\_file in directory:  
 print(f"File: {specific\_file}")  
 print(f"Size: {directory[specific\_file]['size']} bytes")  
 print(f"Created: {directory[specific\_file]['created']}")  
 print(f"Modified: {directory[specific\_file]['modified']}")  
 print(f"Path: {directory[specific\_file]['path']}")  
else:  
 print(f"File {specific\_file} not found in the system")

### ***FINAL Output***



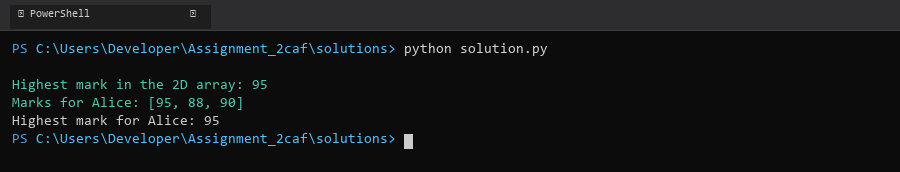
## ***QUESTION 21***

***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 = max(max(row) for row in student\_marks)  
  
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:  
 print(f"Marks for {search\_name}: {student\_data[search\_name]}")  
 print(f"Highest mark for {search\_name}: {max(student\_data[search\_name])}")  
else:  
 print(f"Student {search\_name} not found")

### ***FINAL Output***



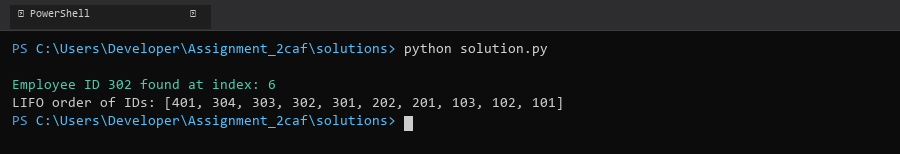
## ***QUESTION 22***

***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  
  
jagged\_array = [  
 [101, 102, 103],  
 [201, 202],  
 [301, 302, 303, 304],  
 [401]  
]  
  
flattened\_array = []  
for subarray in jagged\_array:  
 flattened\_array.extend(sorted(subarray))  
  
stack = Stack()  
for id in flattened\_array:  
 stack.push(id)  
  
target\_id = 302  
search\_result = binary\_search(flattened\_array, target\_id)  
  
print(f"Employee ID {target\_id} found at index: {search\_result}")  
  
popped\_ids = []  
while not stack.is\_empty():  
 popped\_ids.append(stack.pop())  
  
print("LIFO order of IDs:", popped\_ids)

### ***FINAL Output***



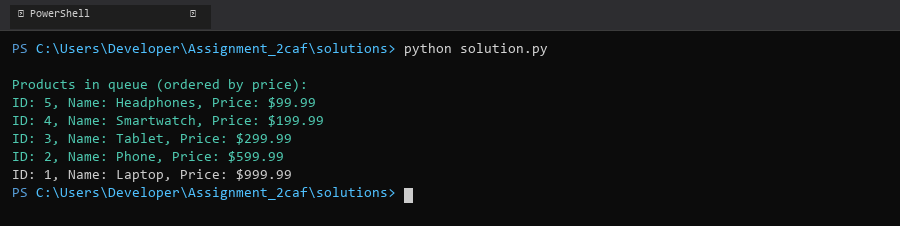
## ***QUESTION 23***

***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  
  
 def \_\_lt\_\_(self, other):  
 return self.price < other.price  
  
from collections import deque  
  
products = [  
 Product(1, "Laptop", 999.99),  
 Product(2, "Phone", 599.99),  
 Product(3, "Tablet", 299.99),  
 Product(4, "Smartwatch", 199.99),  
 Product(5, "Headphones", 99.99)  
]  
  
sorted\_products = sorted(products, key=lambda x: x.price)  
  
product\_queue = deque()  
  
for product in sorted\_products:  
 product\_queue.append(product)  
  
print("Products in queue (ordered by price):")  
while product\_queue:  
 current\_product = product\_queue.popleft()  
 print(f"ID: {current\_product.id}, Name: {current\_product.name}, Price: ${current\_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(f"Sum of diagonal elements: {diagonal\_sum}")

### ***FINAL Output***

