**SOFTWARE ENGENEERING**

**3 bca b**

**"Practical - 4"**

***BY***

**"geetha" (23215134)**

**SUBMITTED TO**

**prateek singh**

****

**SCHOOL OF SCIENCES**

**2025-2026**

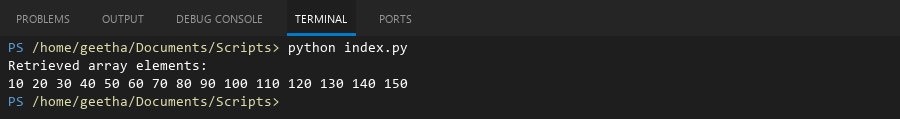
## **QUESTION 1**

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 array as arr  
import pickle  
  
my\_array = arr.array('i', [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150])  
  
with open('array\_data.txt', 'wb') as file:  
 pickle.dump(my\_array, file)  
  
with open('array\_data.txt', 'rb') as file:  
 loaded\_array = pickle.load(file)  
  
print("Retrieved array elements:")  
for element in loaded\_array:  
 print(element, end=" ")

### **FINAL Output**



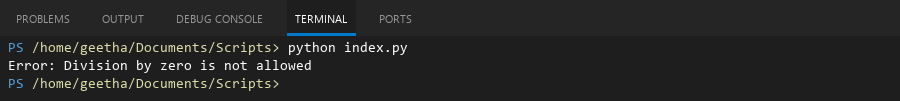
## **QUESTION 2**

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  
 result = num1/num2  
 print(f"Result of division: {result}")  
except ZeroDivisionError:  
 print("Error: Division by zero is not allowed")  
except TypeError:  
 print("Error: Invalid data type for division")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### **FINAL Output**



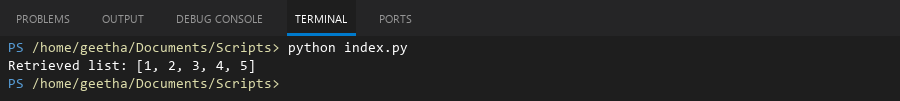
## **QUESTION 3**

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**

numbers = [1, 2, 3, 4, 5]  
  
with open("numbers.txt", "w") as file:  
 for num in numbers:  
 file.write(str(num) + "\n")  
  
stored\_numbers = []  
with open("numbers.txt", "r") as file:  
 for line in file:  
 stored\_numbers.append(int(line.strip()))  
  
print("Retrieved list:", stored\_numbers)

### **FINAL Output**



## **QUESTION 4**

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

### **Code Solution**

def main():  
 numbers = [1, 2, 3, 4, 5]  
 my\_dict = {'a': 1, 'b': 2}  
 my\_string = None  
  
 try:  
 print(numbers[10])  
 except IndexError as e:  
 print(f"Index Error: {e}")  
  
 try:  
 print(my\_string.upper())  
 except AttributeError as e:  
 print(f"Attribute Error: {e}")  
  
 try:  
 value = my\_dict['z']  
 except KeyError as e:  
 print(f"Key Error: {e}")  
  
 try:  
 result = 10 / 0  
 except ZeroDivisionError as e:  
 print(f"Zero Division Error: {e}")  
  
 try:  
 number = int("abc")  
 except ValueError as e:  
 print(f"Value Error: {e}")  
  
 print("Program completed successfully")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

### **FINAL Output**



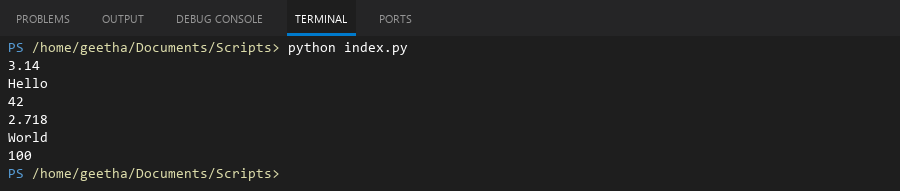
## **QUESTION 5**

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(2.718)  
my\_list.append("World")  
my\_list.append(100)  
  
for item in my\_list:  
 print(item)

### **FINAL Output**



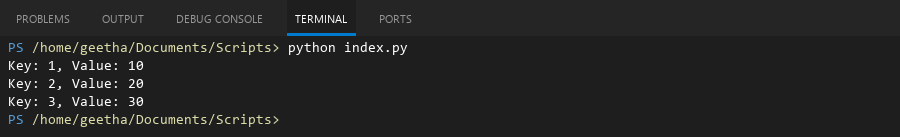
## **QUESTION 6**

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**

hash\_table = {}  
hash\_table[1] = 10  
hash\_table[2] = 20  
hash\_table[3] = 30  
for key, value in hash\_table.items():  
 print(f"Key: {key}, Value: {value}")

### **FINAL Output**



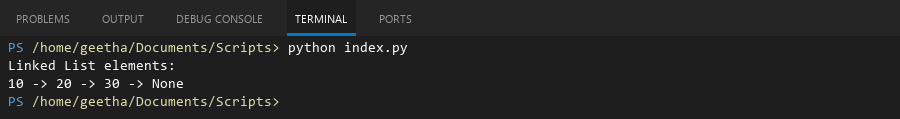
## **QUESTION 7**

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)  
  
print("Linked List elements:")  
linked\_list.print\_list()

### **FINAL Output**



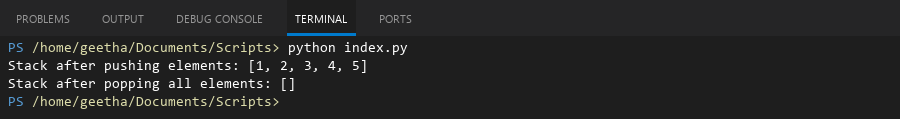
## **QUESTION 8**

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

### **Code Solution**

class Stack:  
 def \_\_init\_\_(self):  
 self.items = []  
   
 def push(self, item):  
 self.items.append(item)  
   
 def pop(self):  
 if not self.is\_empty():  
 return self.items.pop()  
   
 def is\_empty(self):  
 return len(self.items) == 0  
   
 def peek(self):  
 if not self.is\_empty():  
 return self.items[-1]  
   
 def size(self):  
 return len(self.items)  
  
stack = Stack()  
  
stack.push(1)  
stack.push(2)  
stack.push(3)  
stack.push(4)  
stack.push(5)  
  
print("Stack after pushing elements:", stack.items)  
  
for \_ in range(5):  
 stack.pop()  
  
print("Stack after popping all elements:", stack.items)

### **FINAL Output**



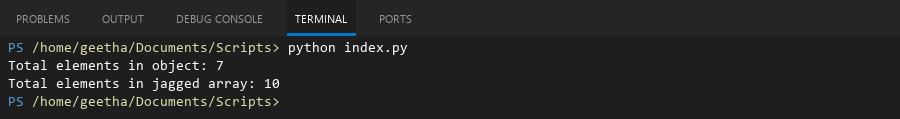
## **QUESTION 9**

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

### **Code Solution**

obj = {'a': 1, 'b': 2, 'c': [1, 2, 3], 'd': {'x': 1, 'y': 2}}  
jagged\_array = [[1, 2, 3], [4, 5], [6, 7, 8, 9], [10]]  
  
def count\_elements\_object(obj):  
 count = 0  
 for value in obj.values():  
 if isinstance(value, dict):  
 count += count\_elements\_object(value)  
 elif isinstance(value, list):  
 count += len(value)  
 else:  
 count += 1  
 return count  
  
def count\_elements\_jagged(arr):  
 count = 0  
 for row in arr:  
 count += len(row)  
 return count  
  
object\_count = count\_elements\_object(obj)  
jagged\_count = count\_elements\_jagged(jagged\_array)  
  
print(f"Total elements in object: {object\_count}")  
print(f"Total elements in jagged array: {jagged\_count}")

### **FINAL Output**



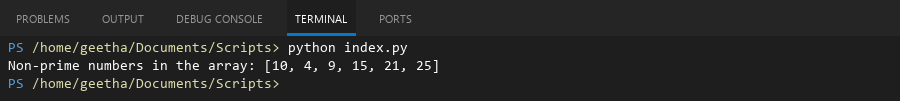
## **QUESTION 10**

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

### **Code Solution**

numbers = [10, 4, 7, 9, 13, 15, 17, 19, 21, 25]  
non\_prime = []  
  
for num in numbers:  
 if num < 2:  
 non\_prime.append(num)  
 continue  
   
 is\_prime = True  
 for i in range(2, int(num \*\* 0.5) + 1):  
 if num % i == 0:  
 is\_prime = False  
 break  
   
 if not is\_prime:  
 non\_prime.append(num)  
  
print("Non-prime numbers in the array:", non\_prime)

### **FINAL Output**



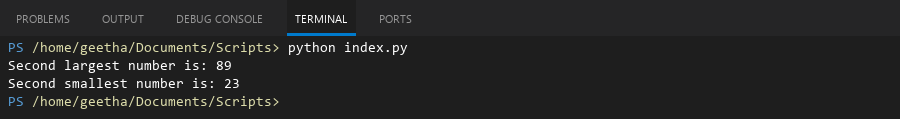
## **QUESTION 11**

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

### **Code Solution**

arr = [45, 32, 89, 23, 67, 12, 90, 34]  
  
if len(arr) < 2:  
 print("Array needs at least 2 elements")  
else:  
 largest = float('-inf')  
 second\_largest = float('-inf')  
 smallest = float('inf')  
 second\_smallest = float('inf')  
  
 for num in arr:  
 if num > largest:  
 second\_largest = largest  
 largest = num  
 elif num > second\_largest and num != largest:  
 second\_largest = num  
   
 if num < smallest:  
 second\_smallest = smallest  
 smallest = num  
 elif num < second\_smallest and num != smallest:  
 second\_smallest = num  
  
 print(f"Second largest number is: {second\_largest}")  
 print(f"Second smallest number is: {second\_smallest}")

### **FINAL Output**



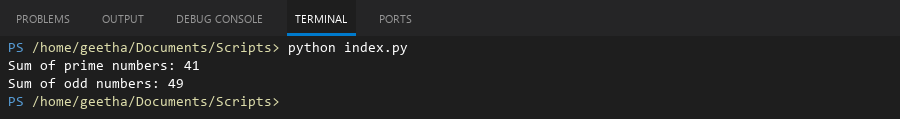
## **QUESTION 12**

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]  
  
prime\_sum = sum(num for num in numbers if is\_prime(num))  
odd\_sum = sum(num for num in numbers if is\_odd(num))  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### **FINAL Output**



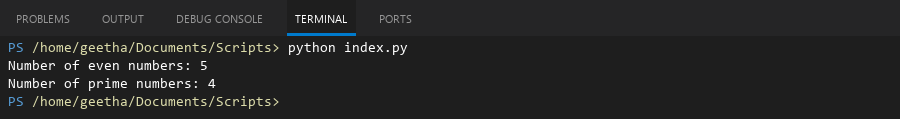
## **QUESTION 13**

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  
  
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
even\_count = 0  
prime\_count = 0  
  
for num in arr:  
 if num % 2 == 0:  
 even\_count += 1  
 if is\_prime(num):  
 prime\_count += 1  
  
print(f"Number of even numbers: {even\_count}")  
print(f"Number of prime numbers: {prime\_count}")

### **FINAL Output**



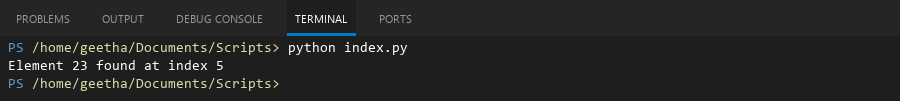
## **QUESTION 14**

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, 50]  
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**



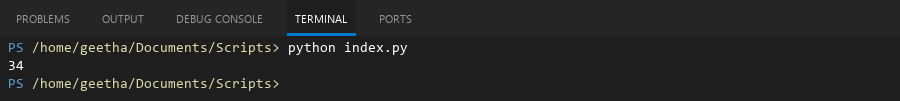
## **QUESTION 15**

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

### **Code Solution**

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

### **FINAL Output**



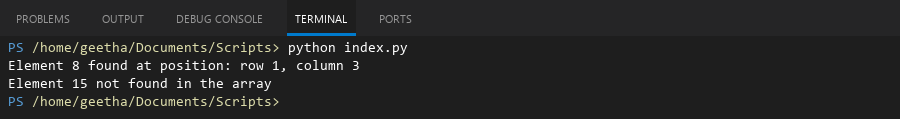
## **QUESTION 16**

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

### **Code Solution**

def linear\_search\_jagged(arr, target):  
 for i in range(len(arr)):  
 for j in range(len(arr[i])):  
 if arr[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  
row, col = linear\_search\_jagged(jagged\_array, target)  
  
if row != -1:  
 print(f"Element {target} found at position: row {row}, column {col}")  
else:  
 print(f"Element {target} not found in the array")  
  
target = 15  
row, col = linear\_search\_jagged(jagged\_array, target)  
  
if row != -1:  
 print(f"Element {target} found at position: row {row}, column {col}")  
else:  
 print(f"Element {target} not found in the array")

### **FINAL Output**



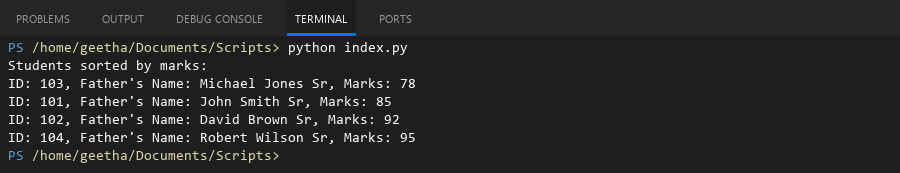
## **QUESTION 17**

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(101, "John Smith Sr", 85),  
 Student(102, "David Brown Sr", 92),  
 Student(103, "Michael Jones Sr", 78),  
 Student(104, "Robert Wilson Sr", 95)  
]  
  
sorted\_students = sorted(students, key=lambda x: x.Marks)  
  
linked\_list = LinkedList()  
for student in sorted\_students:  
 linked\_list.append(student)  
  
print("Students sorted by marks:")  
linked\_list.display()

### **FINAL Output**



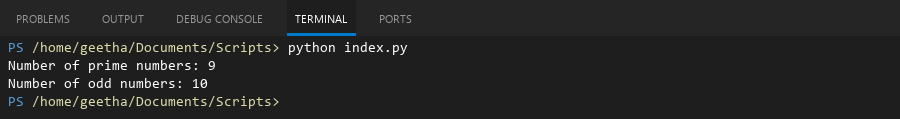
## **QUESTION 18**

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, 19, 23]  
  
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**



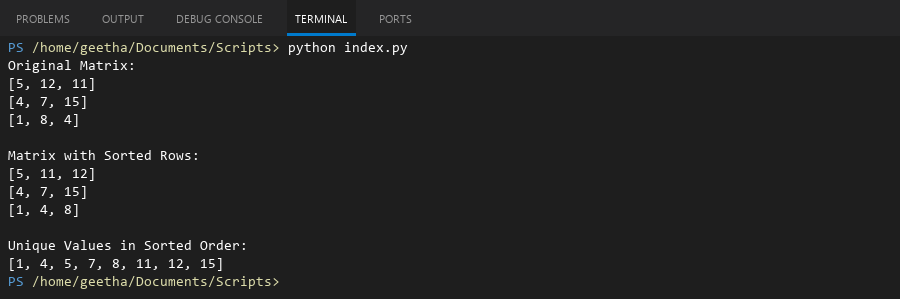
## **QUESTION 19**

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)]  
  
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 Values in Sorted Order:")  
print(unique\_values)

### **FINAL Output**



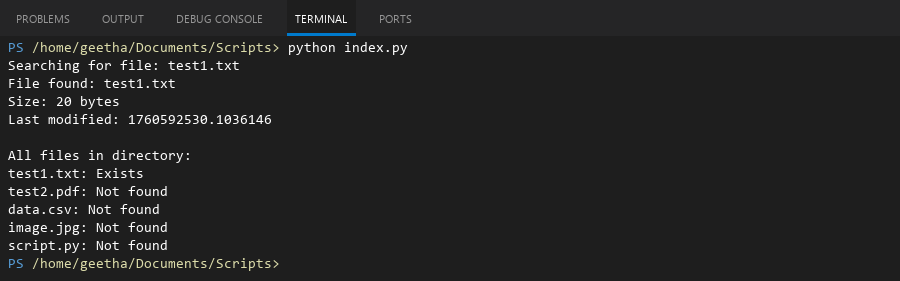
## **QUESTION 20**

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  
  
filenames = ["test1.txt", "test2.pdf", "data.csv", "image.jpg", "script.py"]  
search\_file = "test1.txt"  
directory = defaultdict(dict)  
  
for filename in filenames:  
 if os.path.exists(filename):  
 file\_stats = os.stat(filename)  
 directory[filename] = {  
 "size": file\_stats.st\_size,  
 "modified": file\_stats.st\_mtime,  
 "exists": True  
 }  
 else:  
 directory[filename] = {  
 "exists": False  
 }  
  
print(f"Searching for file: {search\_file}")  
if search\_file in directory:  
 if directory[search\_file]["exists"]:  
 print(f"File found: {search\_file}")  
 print(f"Size: {directory[search\_file]['size']} bytes")  
 print(f"Last modified: {directory[search\_file]['modified']}")  
 else:  
 print(f"File {search\_file} does not exist in the system")  
else:  
 print("File not in the collection")  
  
print("\nAll files in directory:")  
for file, details in directory.items():  
 print(f"{file}: {'Exists' if details['exists'] else 'Not found'}")

### **FINAL Output**



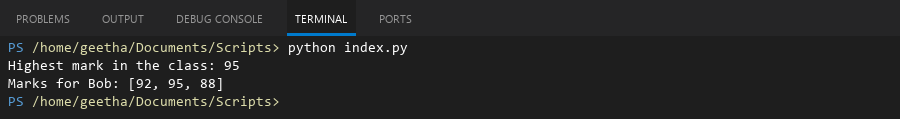
## **QUESTION 21**

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, 90, 92], [78, 88, 85], [92, 95, 88], [75, 82, 80]]  
highest\_mark = max(max(row) for row in student\_marks)  
student\_names = ["John", "Alice", "Bob", "Emma"]  
marks\_dict = {}  
for i in range(len(student\_names)):  
 marks\_dict[student\_names[i]] = student\_marks[i]  
print(f"Highest mark in the class: {highest\_mark}")  
search\_name = "Bob"  
if search\_name in marks\_dict:  
 print(f"Marks for {search\_name}: {marks\_dict[search\_name]}")  
else:  
 print(f"Student {search\_name} not found.")

### **FINAL Output**



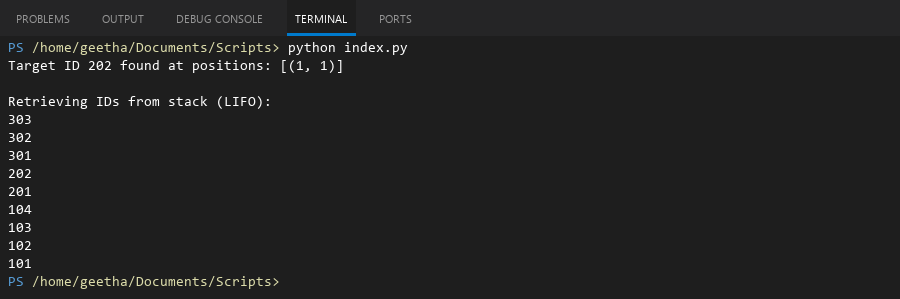
## **QUESTION 22**

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**

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 binary\_search(arr, target):  
 left, right = 0, 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  
  
jagged\_array = [  
 [101, 102, 103, 104],  
 [201, 202],  
 [301, 302, 303],  
]  
  
employee\_stack = Stack()  
target\_id = 202  
  
for row in jagged\_array:  
 row.sort()  
 for id in row:  
 employee\_stack.push(id)  
  
found\_indices = []  
for i, row in enumerate(jagged\_array):  
 result = binary\_search(row, target\_id)  
 if result != -1:  
 found\_indices.append((i, result))  
  
print(f"Target ID {target\_id} found at positions: {found\_indices}")  
  
print("\nRetrieving IDs from stack (LIFO):")  
while not employee\_stack.is\_empty():  
 print(employee\_stack.pop())

### **FINAL Output**



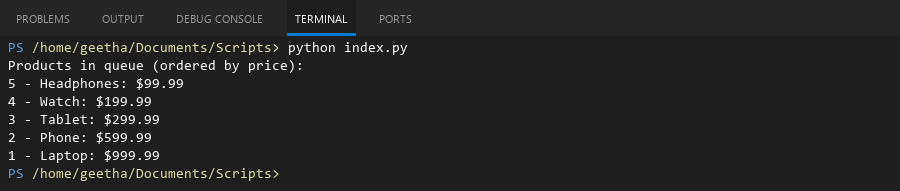
## **QUESTION 23**

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  
  
 def \_\_str\_\_(self):  
 return f"{self.id} - {self.name}: ${self.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", 999.99),  
 Product(2, "Phone", 599.99),  
 Product(3, "Tablet", 299.99),  
 Product(4, "Watch", 199.99),  
 Product(5, "Headphones", 99.99)  
]  
  
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():  
 print(product\_queue.dequeue())

### **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]]  
n = len(matrix)  
diagonal\_sum = 0  
for i in range(n):  
 diagonal\_sum += matrix[i][i]  
print(f"Sum of diagonal elements: {diagonal\_sum}")

### **FINAL Output**

