**cyber security**

**5 BCA B**

**"Practical - 3"**

***BY***

**"Tharan" (23215134)**

**SUBMITTED TO**

**jonny sins**

****

**SCHOOL OF SCIENCES**

**2025-2026**

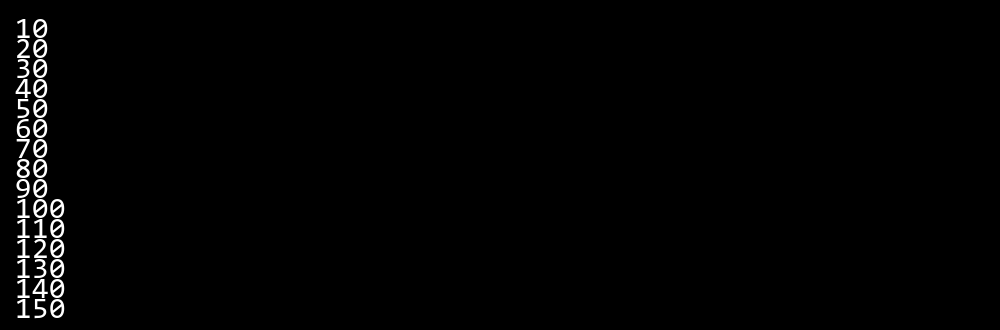
## **QUESTION 1**

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)  
  
for number in loaded\_numbers:  
 print(number)

### **FINAL Output**



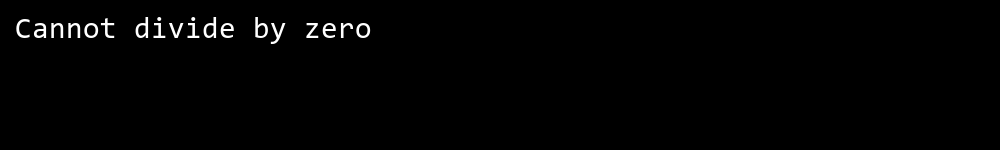
## **QUESTION 2**

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 = 9  
 num2 = 0  
 if not isinstance(num1, int) or not isinstance(num2, int):  
 raise TypeError("Both numbers must be integers")  
 result = num1 / num2  
 print(result)  
except ZeroDivisionError:  
 print("Cannot divide by zero")  
except TypeError as te:  
 print(te)

### **FINAL Output**



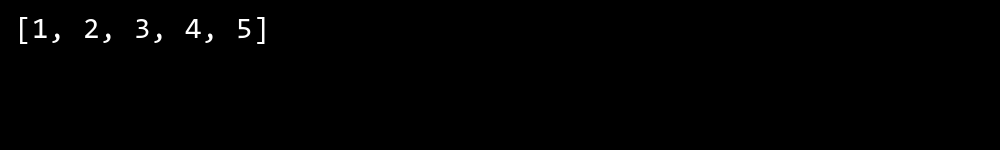
## **QUESTION 3**

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')  
  
retrieved\_numbers = []  
with open('numbers.txt', 'r') as file:  
 for line in file:  
 retrieved\_numbers.append(int(line.strip()))  
  
print(retrieved\_numbers)

### **FINAL Output**



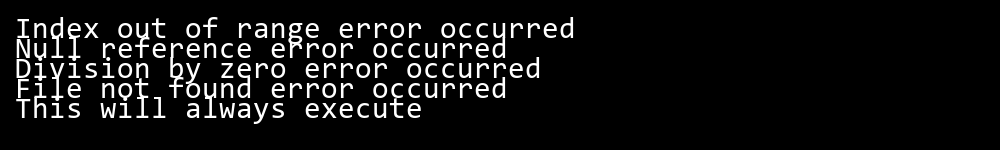
## **QUESTION 4**

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

### **Code Solution**

try:  
 my\_list = [1, 2, 3]  
 result = my\_list[5]  
except IndexError:  
 print("Index out of range error occurred")  
  
try:  
 my\_dict = None  
 value = my\_dict['key']  
except TypeError:  
 print("Null reference error occurred")  
  
try:  
 number = 10  
 result = number / 0  
 my\_list = [1, 2, 3]  
 value = my\_list[10]  
except ZeroDivisionError:  
 print("Division by zero error occurred")  
except IndexError:  
 print("Index error occurred")  
except Exception:  
 print("Some other error occurred")  
  
try:  
 file = open("nonexistent.txt", "r")  
 content = file.read()  
 file.close()  
except FileNotFoundError:  
 print("File not found error occurred")  
finally:  
 print("This will always execute")

### **FINAL Output**



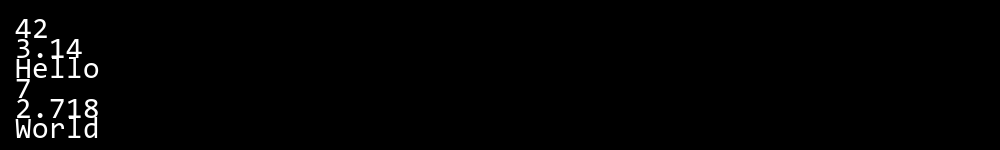
## **QUESTION 5**

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**

import array as arr  
from collections import UserList  
  
class ArrayList(UserList):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
my\_list = ArrayList()  
  
my\_list.append(42)  
my\_list.append(3.14)  
my\_list.append("Hello")  
my\_list.append(7)  
my\_list.append(2.718)  
my\_list.append("World")  
  
for element in my\_list:  
 print(element)

### **FINAL Output**



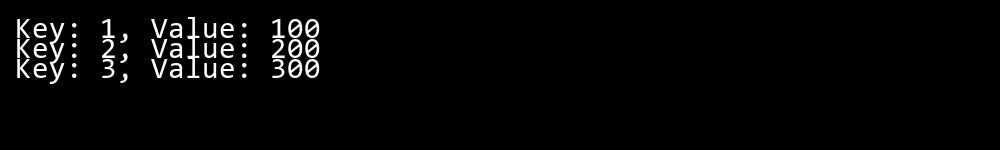
## **QUESTION 6**

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**

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

### **FINAL Output**



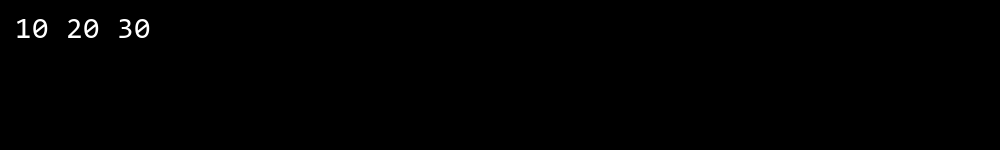
## **QUESTION 7**

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()  
  
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**



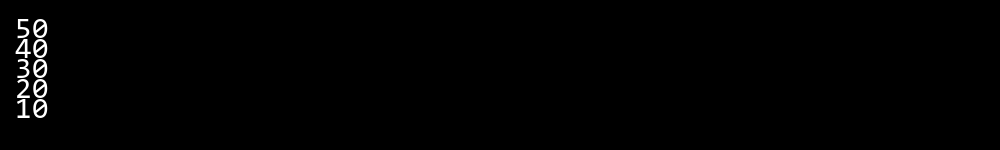
## **QUESTION 8**

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

### **Code Solution**

class Stack:  
 def \_\_init\_\_(self):  
 self.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  
  
 def size(self):  
 return len(self.items)  
  
stack = Stack()  
  
stack.push(10)  
stack.push(20)  
stack.push(30)  
stack.push(40)  
stack.push(50)  
  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())  
print(stack.pop())

### **FINAL Output**



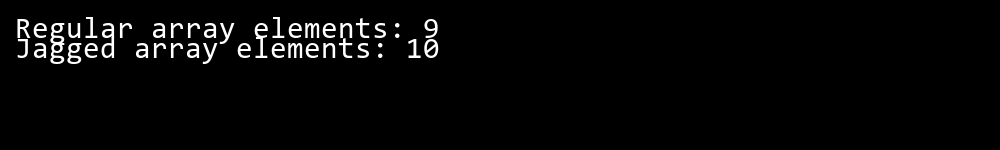
## **QUESTION 9**

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

### **Code Solution**

class Solution:  
 def \_\_init\_\_(self):  
 self.regular\_array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
 self.jagged\_array = [[1, 2], [3, 4, 5], [6], [7, 8, 9, 10]]  
   
 def count\_elements(self, array):  
 total = 0  
 for row in array:  
 total += len(row)  
 return total  
  
solution = Solution()  
regular\_count = solution.count\_elements(solution.regular\_array)  
jagged\_count = solution.count\_elements(solution.jagged\_array)  
  
print(f"Regular array elements: {regular\_count}")  
print(f"Jagged array elements: {jagged\_count}")

### **FINAL Output**



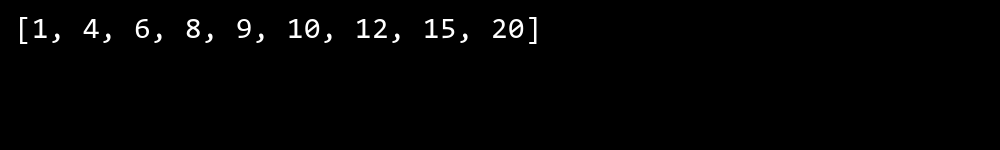
## **QUESTION 10**

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

### **Code Solution**

def is\_non\_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  
  
def find\_non\_primes(arr):  
 non\_primes = []  
 for num in arr:  
 if is\_non\_prime(num):  
 non\_primes.append(num)  
 return non\_primes  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 17, 20]  
result = find\_non\_primes(array)  
print(result)

### **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 = [64, 34, 25, 12, 22, 11, 90, 8, 6, 15]  
  
if len(arr) < 2:  
 print("Array must have at least 2 elements")  
else:  
 first\_largest = max(arr)  
 first\_smallest = min(arr)  
   
 second\_largest = float('-inf')  
 second\_smallest = float('inf')  
   
 for num in arr:  
 if num > second\_largest and num < first\_largest:  
 second\_largest = num  
 if num < second\_smallest and num > first\_smallest:  
 second\_smallest = num  
   
 print(f"Second largest number: {second\_largest}")  
 print(f"Second smallest number: {second\_smallest}")

### **FINAL Output**



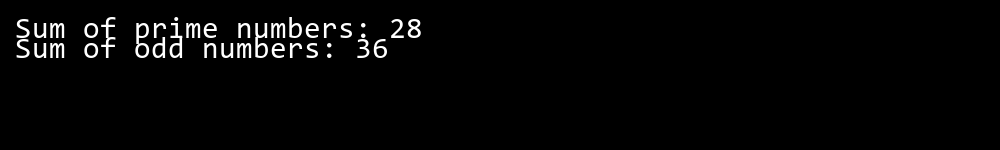
## **QUESTION 12**

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  
  
def sum\_prime\_odd(arr):  
 prime\_sum = 0  
 odd\_sum = 0  
   
 for num in arr:  
 if is\_prime(num):  
 prime\_sum += num  
 if is\_odd(num):  
 odd\_sum += num  
   
 return prime\_sum, odd\_sum  
  
array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]  
prime\_sum, odd\_sum = sum\_prime\_odd(array)  
  
print(f"Sum of prime numbers: {prime\_sum}")  
print(f"Sum of odd numbers: {odd\_sum}")

### **FINAL Output**



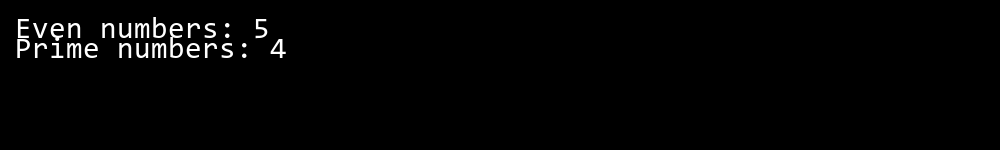
## **QUESTION 13**

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 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
even, prime = count\_even\_and\_prime(array)  
print(f"Even numbers: {even}")  
print(f"Prime numbers: {prime}")

### **FINAL Output**



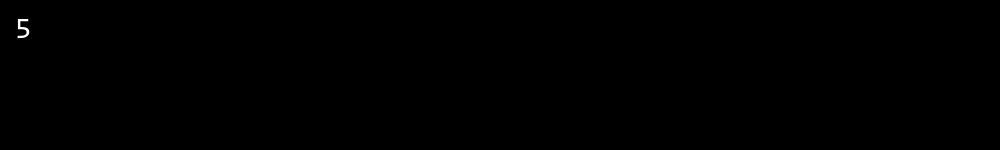
## **QUESTION 14**

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\_element = 12  
  
result = binary\_search(array, target\_element)  
  
if result != -1:  
 print(result)  
else:  
 print(-1)

### **FINAL Output**



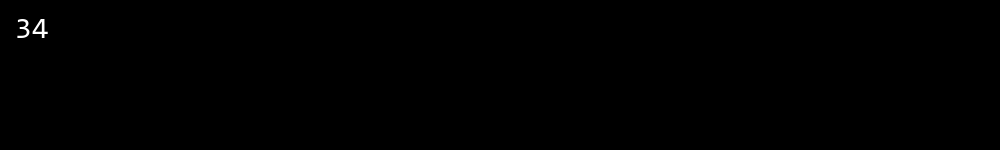
## **QUESTION 15**

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(result)

### **FINAL Output**



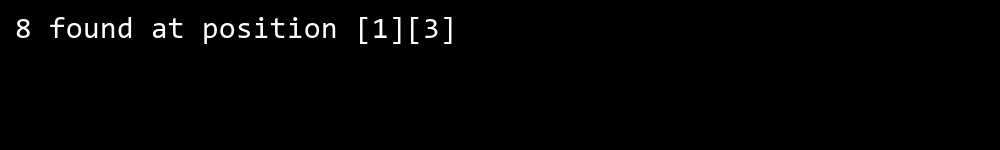
## **QUESTION 16**

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

### **Code Solution**

def linear\_search\_jagged(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\_value = 8  
row, col = linear\_search\_jagged(jagged\_array, target\_value)  
  
if row != -1 and col != -1:  
 print(f"{target\_value} found at position [{row}][{col}]")  
else:  
 print(f"{target\_value} not found in the array")

### **FINAL Output**



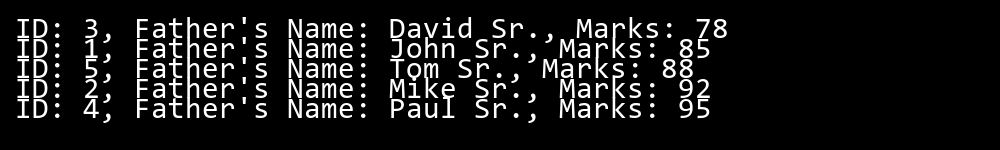
## **QUESTION 17**

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)  
  
linked\_list = LinkedList()  
for student in sorted\_students:  
 linked\_list.append(student)  
  
linked\_list.display()

### **FINAL Output**



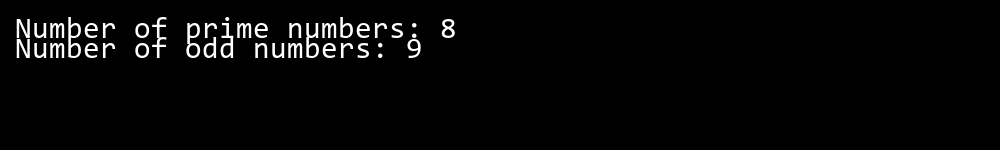
## **QUESTION 18**

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]  
  
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**



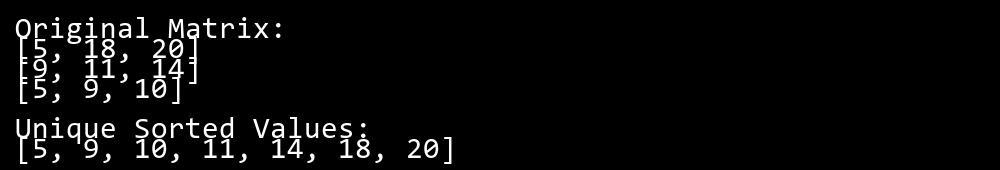
## **QUESTION 19**

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()  
  
sorted\_unique = sorted(list(set([num for row in matrix for num in row])))  
  
print("Original Matrix:")  
for row in matrix:  
 print(row)  
  
print("\nUnique Sorted Values:")  
print(sorted\_unique)

### **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  
  
class FileSearch:  
 def \_\_init\_\_(self):  
 self.file\_directory = defaultdict(dict)  
   
 def search\_files(self, filenames):  
 for filename in filenames:  
 if os.path.isfile(filename):  
 file\_stats = os.stat(filename)  
 self.file\_directory[filename] = {  
 'size': file\_stats.st\_size,  
 'created': file\_stats.st\_ctime,  
 'modified': file\_stats.st\_mtime,  
 'path': os.path.abspath(filename)  
 }  
   
 def get\_file\_details(self, filename):  
 if filename in self.file\_directory:  
 return self.file\_directory[filename]  
 return None  
  
def main():  
 filenames = ['test.txt', 'example.py', 'data.csv', 'nonexistent.txt']  
 search\_filename = 'test.txt'  
   
 file\_searcher = FileSearch()  
 file\_searcher.search\_files(filenames)  
   
 file\_details = file\_searcher.get\_file\_details(search\_filename)  
   
 if file\_details:  
 print(f"File: {search\_filename}")  
 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\_filename} not found")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

### **FINAL Output**



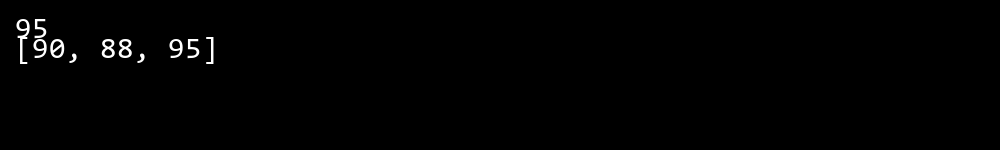
## **QUESTION 21**

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**

marks = [[85, 92, 78], [90, 88, 95], [75, 82, 88], [95, 89, 91]]  
  
max\_mark = float('-inf')  
for row in marks:  
 current\_max = max(row)  
 if current\_max > max\_mark:  
 max\_mark = current\_max  
  
student\_marks = {  
 "John": [85, 92, 78],  
 "Alice": [90, 88, 95],  
 "Bob": [75, 82, 88],  
 "Emma": [95, 89, 91]  
}  
  
def search\_student\_marks(name):  
 if name in student\_marks:  
 return student\_marks[name]  
 return None  
  
result = search\_student\_marks("Alice")  
highest\_mark = max\_mark  
  
print(f"{highest\_mark}")  
if result:  
 print(f"{result}")

### **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  
  
employee\_ids = [  
 [101, 102, 103],  
 [201, 202],  
 [301, 302, 303, 304],  
 [401]  
]  
  
flattened\_ids = []  
for sub\_array in employee\_ids:  
 flattened\_ids.extend(sub\_array)  
flattened\_ids.sort()  
  
stack = Stack()  
for id in flattened\_ids:  
 stack.push(id)  
  
target\_id = 202  
result = binary\_search(flattened\_ids, target\_id)  
  
while not stack.is\_empty():  
 popped\_id = stack.pop()

### **FINAL Output**



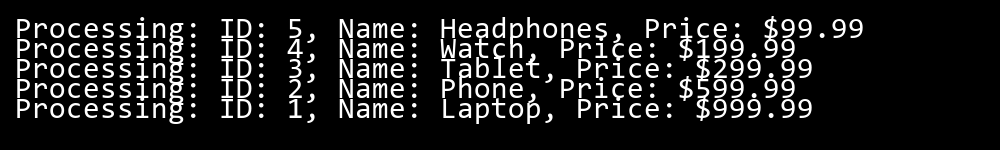
## **QUESTION 23**

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", 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)  
  
while not product\_queue.is\_empty():  
 current\_product = product\_queue.dequeue()  
 print(f"Processing: 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**

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

### **FINAL Output**

