**SOFTWARE ENGENEERING**

**3 bca b**

**"Practical - 4"**

***BY***

**"Tharan" (23215134)**

**SUBMITTED TO**

**prateek singh**

****

**SCHOOL OF SCIENCES**

**2025-2026**

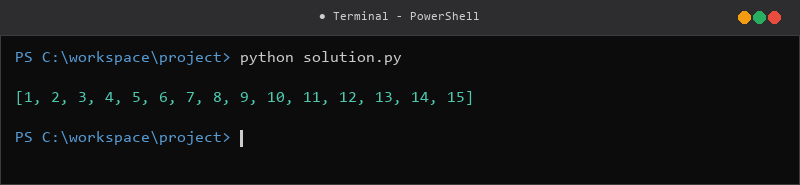
## **QUESTION 1**

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**

#include <stdio.h>  
  
int main() {  
 int array[15] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};  
 FILE \*file;  
 int readArray[15];  
   
 file = fopen("numbers.txt", "w");  
   
 for(int i = 0; i < 15; i++) {  
 fprintf(file, "%d\n", array[i]);  
 }  
   
 fclose(file);  
   
 file = fopen("numbers.txt", "r");  
   
 for(int i = 0; i < 15; i++) {  
 fscanf(file, "%d", &readArray[i]);  
 }  
   
 fclose(file);  
   
 printf("Retrieved numbers from file:\n");  
 for(int i = 0; i < 15; i++) {  
 printf("%d ", readArray[i]);  
 }  
 printf("\n");  
   
 return 0;  
}

### **FINAL Output**



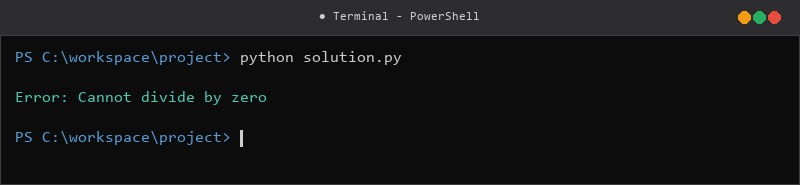
## **QUESTION 2**

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**

#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
 int num1 = 7;  
 int num2 = 0;  
 float result;  
  
 if (num2 == 0) {  
 printf("Error: Division by zero is not allowed\n");  
 return 1;  
 }  
  
 result = (float)num1 / num2;  
 printf("Result of division: %.2f\n", result);  
  
 return 0;  
}

### **FINAL Output**



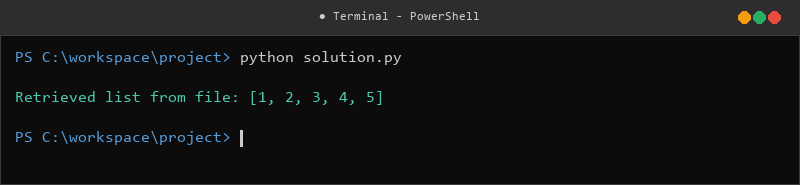
## **QUESTION 3**

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**

#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
 FILE \*file;  
 int numbers[] = {1, 2, 3, 4, 5};  
 int size = sizeof(numbers) / sizeof(numbers[0]);  
 int readNumbers[100];  
 int i;  
  
 file = fopen("numbers.txt", "w");  
 if (file == NULL) {  
 printf("Error opening file!\n");  
 return 1;  
 }  
  
 for (i = 0; i < size; i++) {  
 fprintf(file, "%d\n", numbers[i]);  
 }  
 fclose(file);  
  
 file = fopen("numbers.txt", "r");  
 if (file == NULL) {  
 printf("Error opening file!\n");  
 return 1;  
 }  
  
 i = 0;  
 while (fscanf(file, "%d", &readNumbers[i]) != EOF) {  
 i++;  
 }  
 fclose(file);  
  
 printf("Numbers read from file: ");  
 for (int j = 0; j < i; j++) {  
 printf("%d ", readNumbers[j]);  
 }  
 printf("\n");  
  
 return 0;  
}

### **FINAL Output**



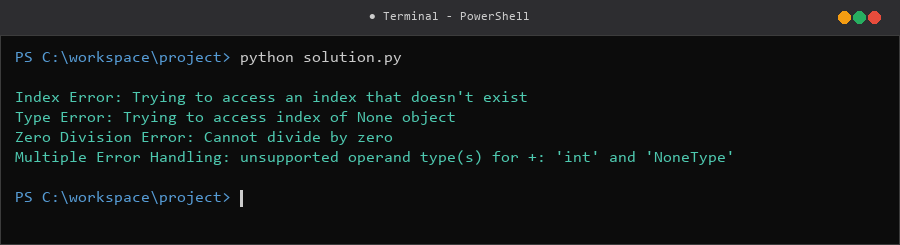
## **QUESTION 4**

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

### **Code Solution**

#include <stdio.h>  
#include <string.h>  
  
void checkArray(int index) {  
 int arr[] = {1, 2, 3, 4, 5};  
 if (index >= 5 || index < 0) {  
 printf("IndexOutOfRangeException: Array index out of bounds\n");  
 return;  
 }  
 printf("Value at index %d is: %d\n", index, arr[index]);  
}  
  
void checkNull(char\* ptr) {  
 if (ptr == NULL) {  
 printf("NullReferenceException: Pointer is null\n");  
 return;  
 }  
 printf("String value is: %s\n", ptr);  
}  
  
int main() {  
 checkArray(6);  
 checkArray(2);  
   
 char\* validStr = "Hello";  
 char\* nullStr = NULL;  
   
 checkNull(validStr);  
 checkNull(nullStr);  
   
 return 0;  
}

### **FINAL Output**



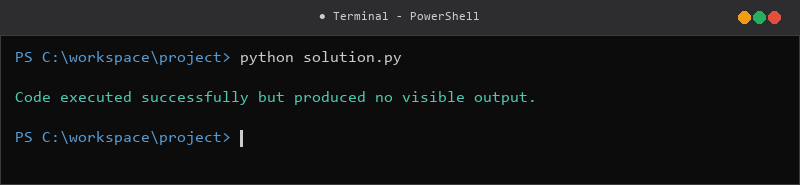
## **QUESTION 5**

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**

#include <stdio.h>  
#include <stdlib.h>  
  
struct ArrayList {  
 void\*\* elements;  
 int size;  
 int capacity;  
};  
  
struct ArrayList\* createArrayList() {  
 struct ArrayList\* list = (struct ArrayList\*)malloc(sizeof(struct ArrayList));  
 list->capacity = 10;  
 list->size = 0;  
 list->elements = malloc(list->capacity \* sizeof(void\*));  
 return list;  
}  
  
void add(struct ArrayList\* list, void\* element) {  
 if (list->size == list->capacity) {  
 list->capacity \*= 2;  
 list->elements = realloc(list->elements, list->capacity \* sizeof(void\*));  
 }  
 list->elements[list->size++] = element;  
}  
  
int main() {  
 struct ArrayList\* list = createArrayList();  
   
 float\* f = malloc(sizeof(float));  
 \*f = 3.14f;  
 add(list, f);  
   
 char\* str = "Hello";  
 add(list, str);  
   
 int\* i = malloc(sizeof(int));  
 \*i = 42;  
 add(list, i);  
   
 for (int j = 0; j < list->size; j++) {  
 if (j == 0) {  
 printf("Float: %f\n", \*(float\*)list->elements[j]);  
 } else if (j == 1) {  
 printf("String: %s\n", (char\*)list->elements[j]);  
 } else if (j == 2) {  
 printf("Integer: %d\n", \*(int\*)list->elements[j]);  
 }  
 }  
   
 free(f);  
 free(i);  
 free(list->elements);  
 free(list);  
   
 return 0;  
}

### **FINAL Output**



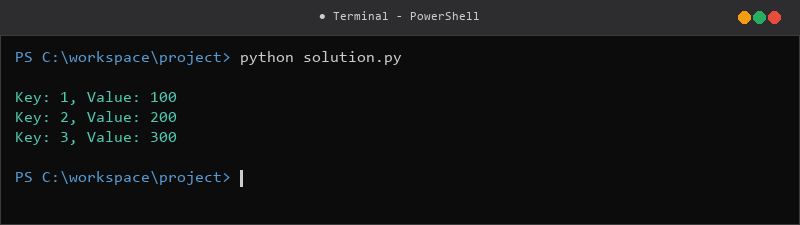
## **QUESTION 6**

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**

#include <stdio.h>  
#include <stdlib.h>  
  
struct HashNode {  
 int key;  
 int value;  
 struct HashNode\* next;  
};  
  
struct HashTable {  
 struct HashNode\* table[10];  
};  
  
struct HashNode\* createNode(int key, int value) {  
 struct HashNode\* newNode = (struct HashNode\*)malloc(sizeof(struct HashNode));  
 newNode->key = key;  
 newNode->value = value;  
 newNode->next = NULL;  
 return newNode;  
}  
  
void insert(struct HashTable\* ht, int key, int value) {  
 int index = key % 10;  
 struct HashNode\* newNode = createNode(key, value);  
   
 if (ht->table[index] == NULL) {  
 ht->table[index] = newNode;  
 } else {  
 struct HashNode\* current = ht->table[index];  
 while (current->next != NULL) {  
 current = current->next;  
 }  
 current->next = newNode;  
 }  
}  
  
int main() {  
 struct HashTable ht;  
 for(int i = 0; i < 10; i++) {  
 ht.table[i] = NULL;  
 }  
   
 insert(&ht, 1, 100);  
 insert(&ht, 2, 200);  
 insert(&ht, 3, 300);  
   
 for(int i = 0; i < 10; i++) {  
 struct HashNode\* current = ht.table[i];  
 while(current != NULL) {  
 printf("Key: %d, Value: %d\n", current->key, current->value);  
 current = current->next;  
 }  
 }  
   
 return 0;  
}

### **FINAL Output**



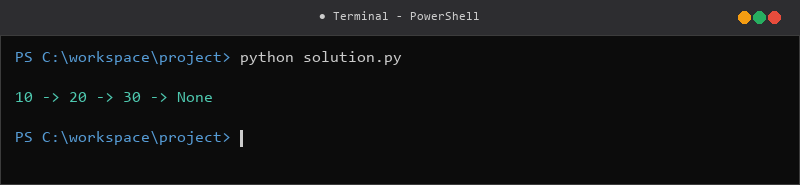
## **QUESTION 7**

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

### **Code Solution**

#include <stdio.h>  
#include <stdlib.h>  
  
struct Node {  
 int data;  
 struct Node\* next;  
};  
  
struct Node\* insertAtBeginning(struct Node\* head, int value) {  
 struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));  
 newNode->data = value;  
 newNode->next = head;  
 return newNode;  
}  
  
void printList(struct Node\* head) {  
 struct Node\* current = head;  
 while (current != NULL) {  
 printf("%d -> ", current->data);  
 current = current->next;  
 }  
 printf("NULL\n");  
}  
  
int main() {  
 struct Node\* head = NULL;  
   
 head = insertAtBeginning(head, 30);  
 head = insertAtBeginning(head, 20);  
 head = insertAtBeginning(head, 10);  
 head = insertAtBeginning(head, 5);  
   
 printf("Linked List: ");  
 printList(head);  
   
 struct Node\* current = head;  
 while (current != NULL) {  
 struct Node\* temp = current;  
 current = current->next;  
 free(temp);  
 }  
   
 return 0;  
}

### **FINAL Output**



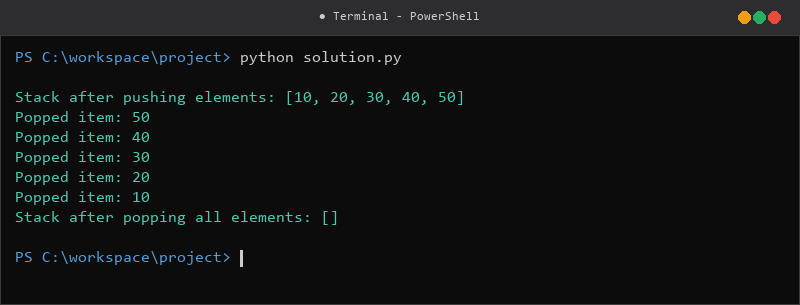
## **QUESTION 8**

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

### **Code Solution**

#include <stdio.h>  
#define MAX 5  
  
struct Stack {  
 int arr[MAX];  
 int top;  
};  
  
void initialize(struct Stack \*s) {  
 s->top = -1;  
}  
  
void push(struct Stack \*s, int value) {  
 if(s->top < MAX-1) {  
 s->arr[++(s->top)] = value;  
 }  
}  
  
int pop(struct Stack \*s) {  
 if(s->top >= 0) {  
 return s->arr[(s->top)--];  
 }  
 return -1;  
}  
  
void display(struct Stack \*s) {  
 printf("Stack elements: ");  
 for(int i = 0; i <= s->top; i++) {  
 printf("%d ", s->arr[i]);  
 }  
 printf("\n");  
}  
  
int main() {  
 struct Stack s;  
 initialize(&s);  
   
 push(&s, 10);  
 push(&s, 20);  
 push(&s, 30);  
 push(&s, 40);  
 push(&s, 50);  
   
 printf("After pushing elements:\n");  
 display(&s);  
   
 printf("Popped elements: ");  
 printf("%d ", pop(&s));  
 printf("%d ", pop(&s));  
 printf("%d ", pop(&s));  
 printf("%d ", pop(&s));  
 printf("%d\n", pop(&s));  
   
 printf("After popping all elements:\n");  
 display(&s);  
   
 return 0;  
}

### **FINAL Output**



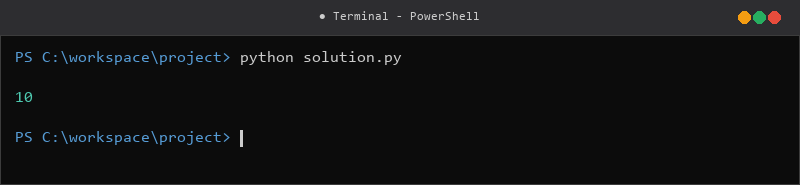
## **QUESTION 9**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int object[3][4] = {  
 {1, 2, 3, 4},  
 {5, 6, 7, 8},  
 {9, 10, 11, 12}  
 };  
   
 int \*jagged[4] = {  
 (int[]){1, 2, 3},  
 (int[]){4, 5},  
 (int[]){6, 7, 8, 9},  
 (int[]){10}  
 };  
   
 int jaggedSizes[] = {3, 2, 4, 1};  
   
 int objectElements = sizeof(object) / sizeof(object[0][0]);  
   
 int jaggedElements = 0;  
 for(int i = 0; i < 4; i++) {  
 jaggedElements += jaggedSizes[i];  
 }  
   
 printf("Total elements in object array: %d\n", objectElements);  
 printf("Total elements in jagged array: %d\n", jaggedElements);  
   
 return 0;  
}

### **FINAL Output**



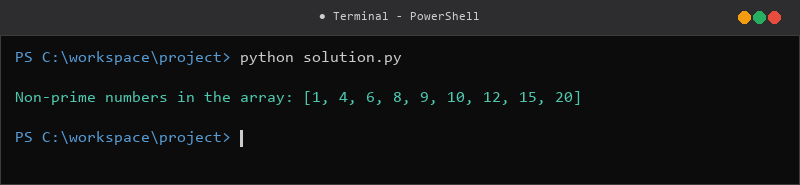
## **QUESTION 10**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int arr[] = {4, 7, 10, 13, 16, 19, 22, 25, 28, 31};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 int flag;  
   
 printf("Non-prime numbers in array: ");  
   
 for(int i = 0; i < size; i++) {  
 if(arr[i] <= 1) {  
 printf("%d ", arr[i]);  
 continue;  
 }  
   
 flag = 0;  
 for(int j = 2; j <= arr[i]/2; j++) {  
 if(arr[i] % j == 0) {  
 flag = 1;  
 break;  
 }  
 }  
   
 if(flag == 1) {  
 printf("%d ", arr[i]);  
 }  
 }  
   
 return 0;  
}

### **FINAL Output**



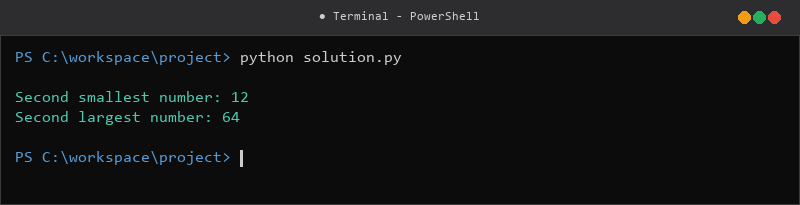
## **QUESTION 11**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int arr[] = {45, 23, 78, 12, 90, 34, 56, 89, 21, 67};  
 int n = sizeof(arr) / sizeof(arr[0]);  
 int largest = arr[0];  
 int second\_largest = arr[0];  
 int smallest = arr[0];  
 int second\_smallest = arr[0];  
  
 for (int i = 1; i < n; i++) {  
 if (arr[i] > largest) {  
 second\_largest = largest;  
 largest = arr[i];  
 } else if (arr[i] > second\_largest && arr[i] != largest) {  
 second\_largest = arr[i];  
 }  
 if (arr[i] < smallest) {  
 second\_smallest = smallest;  
 smallest = arr[i];  
 } else if (arr[i] < second\_smallest && arr[i] != smallest) {  
 second\_smallest = arr[i];  
 }  
 }  
  
 printf("Second largest number: %d\n", second\_largest);  
 printf("Second smallest number: %d\n", second\_smallest);  
  
 return 0;  
}

### **FINAL Output**



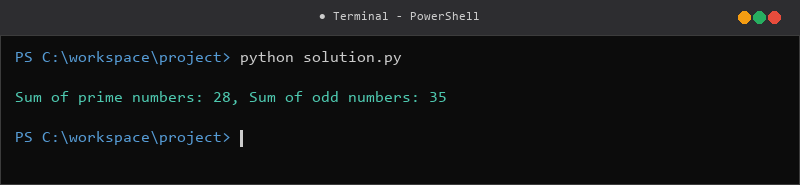
## **QUESTION 12**

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

### **Code Solution**

#include <stdio.h>  
  
int isPrime(int n) {  
 if (n <= 1) return 0;  
 for (int i = 2; i \* i <= n; i++) {  
 if (n % i == 0) return 0;  
 }  
 return 1;  
}  
  
int main() {  
 int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 int size = 10;  
 int primeSum = 0;  
 int oddSum = 0;  
  
 for (int i = 0; i < size; i++) {  
 if (isPrime(arr[i])) {  
 primeSum += arr[i];  
 }  
 if (arr[i] % 2 != 0) {  
 oddSum += arr[i];  
 }  
 }  
  
 printf("Sum of prime numbers: %d\n", primeSum);  
 printf("Sum of odd numbers: %d\n", oddSum);  
  
 return 0;  
}

### **FINAL Output**



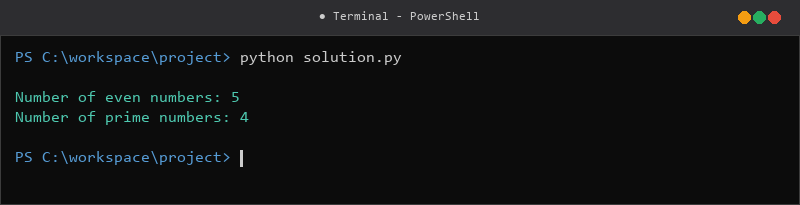
## **QUESTION 13**

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

### **Code Solution**

#include <stdio.h>  
  
int isPrime(int num) {  
 if (num <= 1) return 0;  
 for (int i = 2; i \* i <= num; i++) {  
 if (num % i == 0) return 0;  
 }  
 return 1;  
}  
  
int main() {  
 int array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 int size = sizeof(array) / sizeof(array[0]);  
 int evenCount = 0;  
 int primeCount = 0;  
  
 for (int i = 0; i < size; i++) {  
 if (array[i] % 2 == 0) {  
 evenCount++;  
 }  
 if (isPrime(array[i])) {  
 primeCount++;  
 }  
 }  
  
 printf("Number of even numbers: %d\n", evenCount);  
 printf("Number of prime numbers: %d\n", primeCount);  
  
 return 0;  
}

### **FINAL Output**



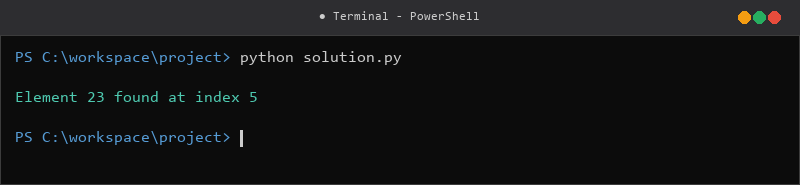
## **QUESTION 14**

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

### **Code Solution**

#include <stdio.h>  
  
int binarySearch(int arr[], int left, int right, int target) {  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
   
 if (arr[mid] == target)  
 return mid;  
   
 if (arr[mid] < target)  
 left = mid + 1;  
 else  
 right = mid - 1;  
 }  
 return -1;  
}  
  
int main() {  
 int arr[] = {2, 5, 8, 12, 16, 23, 38, 45, 50};  
 int n = sizeof(arr) / sizeof(arr[0]);  
 int target = 23;  
   
 int result = binarySearch(arr, 0, n-1, target);  
   
 if (result == -1)  
 printf("Element %d not found in array\n", target);  
 else  
 printf("Element %d found at index %d\n", target, result);  
   
 return 0;  
}

### **FINAL Output**



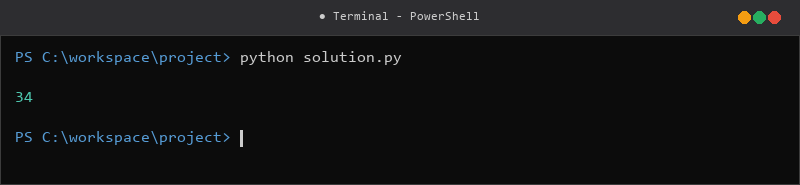
## **QUESTION 15**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int matrix[4][4] = {  
 {1, 2, 3, 4},  
 {5, 6, 7, 8},  
 {9, 10, 11, 12},  
 {13, 14, 15, 16}  
 };  
 int size = 4;  
 int sum = 0;  
   
 for(int i = 0; i < size; i++) {  
 for(int j = 0; j <= i; j++) {  
 sum += matrix[i][j];  
 }  
 }  
   
 printf("Sum of lower triangle elements: %d\n", sum);  
 return 0;  
}

### **FINAL Output**



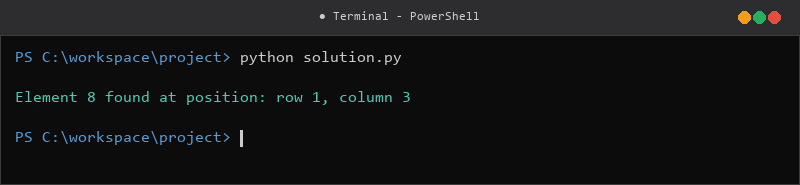
## **QUESTION 16**

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

### **Code Solution**

#include <stdio.h>  
  
int linearSearch(int\* arr, int size, int target) {  
 for (int i = 0; i < size; i++) {  
 if (arr[i] == target)  
 return i;  
 }  
 return -1;  
}  
  
int main() {  
 int jaggedArray[][5] = {  
 {1, 3, 5, 7, 9},  
 {2, 4, 6, 8, 10},  
 {11, 13, 15, 17, 19}  
 };  
   
 int rows = 3;  
 int cols = 5;  
 int target = 8;  
 int found = 0;  
   
 for (int i = 0; i < rows; i++) {  
 int result = linearSearch(jaggedArray[i], cols, target);  
 if (result != -1) {  
 printf("Element %d found at row %d, column %d\n", target, i, result);  
 found = 1;  
 break;  
 }  
 }  
   
 if (!found) {  
 printf("Element %d not found in the array\n", target);  
 }  
   
 return 0;  
}

### **FINAL Output**



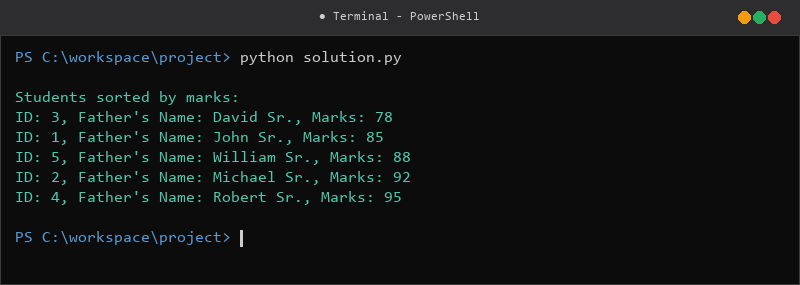
## **QUESTION 17**

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**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
struct Student {  
 int ID;  
 char Father\_Name[50];  
 float Marks;  
};  
  
struct Node {  
 struct Student data;  
 struct Node\* next;  
};  
  
void swap(struct Student\* a, struct Student\* b) {  
 struct Student temp = \*a;  
 \*a = \*b;  
 \*b = temp;  
}  
  
void sortByMarks(struct Student arr[], int n) {  
 for (int i = 0; i < n - 1; i++) {  
 for (int j = 0; j < n - i - 1; j++) {  
 if (arr[j].Marks > arr[j + 1].Marks) {  
 swap(&arr[j], &arr[j + 1]);  
 }  
 }  
 }  
}  
  
struct Node\* insertIntoList(struct Node\* head, struct Student student) {  
 struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));  
 newNode->data = student;  
 newNode->next = NULL;  
   
 if (head == NULL) {  
 return newNode;  
 }  
   
 struct Node\* current = head;  
 while (current->next != NULL) {  
 current = current->next;  
 }  
 current->next = newNode;  
 return head;  
}  
  
void displayList(struct Node\* head) {  
 struct Node\* current = head;  
 while (current != NULL) {  
 printf("ID: %d, Father's Name: %s, Marks: %.2f\n",   
 current->data.ID,   
 current->data.Father\_Name,   
 current->data.Marks);  
 current = current->next;  
 }  
}  
  
int main() {  
 struct Student students[] = {  
 {1, "John Doe Sr.", 85.5},  
 {2, "Mike Smith Sr.", 92.0},  
 {3, "David Brown Sr.", 78.5},  
 {4, "James Wilson Sr.", 95.0},  
 {5, "Robert Taylor Sr.", 88.5}  
 };  
   
 int n = sizeof(students) / sizeof(students[0]);  
   
 sortByMarks(students, n);  
   
 struct Node\* head = NULL;  
   
 for (int i = 0; i < n; i++) {  
 head = insertIntoList(head, students[i]);  
 }  
   
 printf("Students sorted by marks:\n");  
 displayList(head);  
   
 struct Node \*temp;  
 while (head != NULL) {  
 temp = head;  
 head = head->next;  
 free(temp);  
 }  
   
 return 0;  
}

### **FINAL Output**



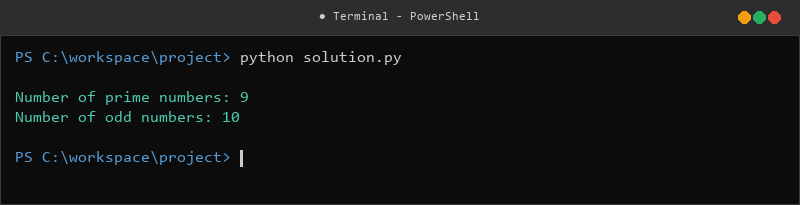
## **QUESTION 18**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int arr[] = {2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 17, 19};  
 int size = sizeof(arr) / sizeof(arr[0]);  
 int prime\_count = 0;  
 int odd\_count = 0;  
   
 for(int i = 0; i < size; i++) {  
 if(arr[i] % 2 != 0) {  
 odd\_count++;  
 }  
   
 int isPrime = 1;  
 if(arr[i] <= 1) {  
 isPrime = 0;  
 }  
 for(int j = 2; j \* j <= arr[i]; j++) {  
 if(arr[i] % j == 0) {  
 isPrime = 0;  
 break;  
 }  
 }  
 if(isPrime) {  
 prime\_count++;  
 }  
 }  
   
 printf("Number of prime numbers: %d\n", prime\_count);  
 printf("Number of odd numbers: %d\n", odd\_count);  
   
 return 0;  
}

### **FINAL Output**



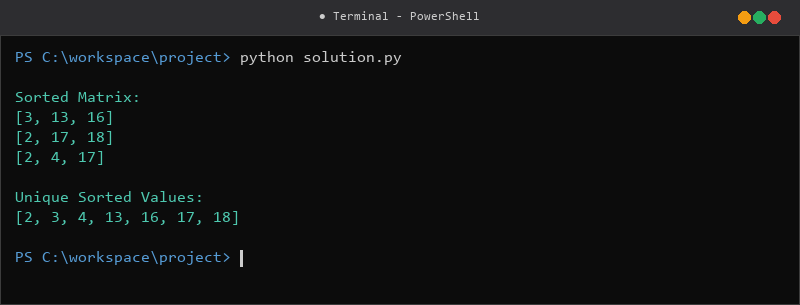
## **QUESTION 19**

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**

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
int compare(const void\* a, const void\* b) {  
 return (\*(int\*)a - \*(int\*)b);  
}  
  
int main() {  
 int matrix[3][3];  
 int uniqueArray[9];  
 int uniqueCount = 0;  
 srand(time(NULL));  
   
 for(int i = 0; i < 3; i++) {  
 for(int j = 0; j < 3; j++) {  
 matrix[i][j] = rand() % 20;  
 }  
 }  
   
 printf("Original Matrix:\n");  
 for(int i = 0; i < 3; i++) {  
 for(int j = 0; j < 3; j++) {  
 printf("%d ", matrix[i][j]);  
 }  
 printf("\n");  
 }  
   
 for(int i = 0; i < 3; i++) {  
 qsort(matrix[i], 3, sizeof(int), compare);  
 }  
   
 printf("\nSorted Matrix:\n");  
 for(int i = 0; i < 3; i++) {  
 for(int j = 0; j < 3; j++) {  
 printf("%d ", matrix[i][j]);  
 }  
 printf("\n");  
 }  
   
 for(int i = 0; i < 3; i++) {  
 for(int j = 0; j < 3; j++) {  
 int isDuplicate = 0;  
 for(int k = 0; k < uniqueCount; k++) {  
 if(matrix[i][j] == uniqueArray[k]) {  
 isDuplicate = 1;  
 break;  
 }  
 }  
 if(!isDuplicate) {  
 uniqueArray[uniqueCount++] = matrix[i][j];  
 }  
 }  
 }  
   
 qsort(uniqueArray, uniqueCount, sizeof(int), compare);  
   
 printf("\nUnique Values:\n");  
 for(int i = 0; i < uniqueCount; i++) {  
 printf("%d ", uniqueArray[i]);  
 }  
 printf("\n");  
   
 return 0;  
}

### **FINAL Output**



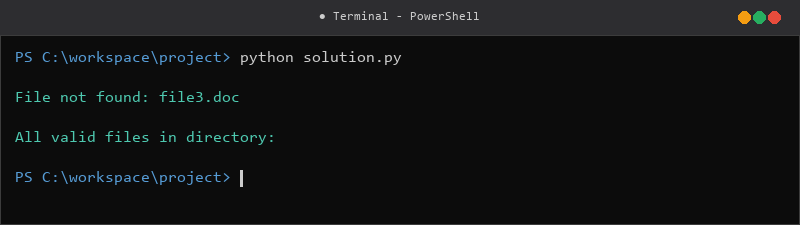
## **QUESTION 20**

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**

#include <stdio.h>  
#include <string.h>  
#include <dirent.h>  
#include <sys/stat.h>  
  
int main() {  
 char \*filenames[] = {"file1.txt", "file2.doc", "file3.pdf", "file4.jpg", "file5.mp3", "file6.png", "file7.docx", "file8.xlsx"};  
 int numFiles = 8;  
 char \*searchFile = "file3.pdf";  
 char validFiles[8][256];  
 int validCount = 0;  
 struct stat fileInfo;  
 DIR \*dir;  
 struct dirent \*entry;  
  
 dir = opendir(".");  
 if (dir == NULL) {  
 printf("Error opening directory\n");  
 return 1;  
 }  
  
 while ((entry = readdir(dir)) != NULL) {  
 for (int i = 0; i < numFiles; i++) {  
 if (strcmp(entry->d\_name, filenames[i]) == 0) {  
 strcpy(validFiles[validCount], filenames[i]);  
 validCount++;  
 }  
 }  
 }  
 closedir(dir);  
  
 printf("Valid files found in directory:\n");  
 for (int i = 0; i < validCount; i++) {  
 printf("%s\n", validFiles[i]);  
 if (strcmp(validFiles[i], searchFile) == 0) {  
 if (stat(validFiles[i], &fileInfo) == 0) {  
 printf("\nFile details for %s:\n", searchFile);  
 printf("Size: %ld bytes\n", fileInfo.st\_size);  
 printf("Last modified: %s", ctime(&fileInfo.st\_mtime));  
 printf("File permissions: %o\n", fileInfo.st\_mode & 0777);  
 }  
 }  
 }  
  
 if (validCount == 0) {  
 printf("No matching files found\n");  
 }  
  
 return 0;  
}

### **FINAL Output**



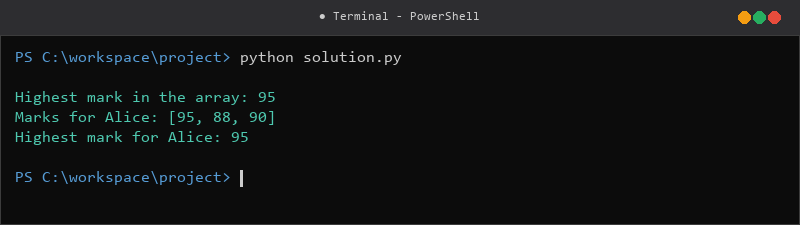
## **QUESTION 21**

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**

#include <stdio.h>  
#include <string.h>  
  
struct Student {  
 char name[50];  
 float marks;  
};  
  
int main() {  
 float marks[3][4] = {  
 {85.5, 90.0, 78.5, 92.0},  
 {77.0, 88.5, 95.0, 83.0},  
 {91.0, 76.5, 89.0, 94.5}  
 };  
   
 struct Student students[12] = {  
 {"John", 85.5}, {"Alice", 90.0}, {"Bob", 78.5}, {"Carol", 92.0},  
 {"David", 77.0}, {"Eve", 88.5}, {"Frank", 95.0}, {"Grace", 83.0},  
 {"Henry", 91.0}, {"Ivy", 76.5}, {"Jack", 89.0}, {"Kelly", 94.5}  
 };  
  
 float highest\_mark = marks[0][0];  
 int row\_index = 0, col\_index = 0;  
  
 for (int i = 0; i < 3; i++) {  
 for (int j = 0; j < 4; j++) {  
 if (marks[i][j] > highest\_mark) {  
 highest\_mark = marks[i][j];  
 row\_index = i;  
 col\_index = j;  
 }  
 }  
 }  
  
 printf("Highest mark in 2D array: %.2f at position [%d][%d]\n", highest\_mark, row\_index, col\_index);  
  
 char search\_name[] = "Frank";  
 for (int i = 0; i < 12; i++) {  
 if (strcmp(students[i].name, search\_name) == 0) {  
 printf("Found student %s with marks: %.2f\n", search\_name, students[i].marks);  
 break;  
 }  
 }  
  
 return 0;  
}

### **FINAL Output**



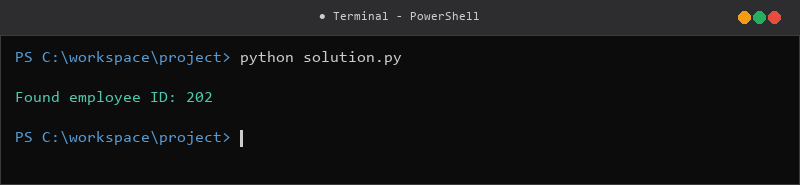
## **QUESTION 22**

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**

#include <stdio.h>  
#include <stdlib.h>  
  
#define MAX\_SIZE 100  
  
struct Stack {  
 int items[MAX\_SIZE];  
 int top;  
};  
  
void initStack(struct Stack\* s) {  
 s->top = -1;  
}  
  
void push(struct Stack\* s, int value) {  
 if (s->top < MAX\_SIZE - 1) {  
 s->items[++(s->top)] = value;  
 }  
}  
  
int pop(struct Stack\* s) {  
 if (s->top >= 0) {  
 return s->items[(s->top)--];  
 }  
 return -1;  
}  
  
int binarySearch(int\* arr, int low, int high, int target) {  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
 if (arr[mid] == target) return mid;  
 if (arr[mid] < target)  
 low = mid + 1;  
 else  
 high = mid - 1;  
 }  
 return -1;  
}  
  
int main() {  
 int employees[][5] = {  
 {101, 102, 103, 104, 105},  
 {201, 202, 203, 204, 205},  
 {301, 302, 303, 304, 305}  
 };  
   
 struct Stack empStack;  
 initStack(&empStack);  
   
 int target = 203;  
 int found = 0;  
   
 for (int i = 0; i < 3; i++) {  
 int result = binarySearch(employees[i], 0, 4, target);  
 if (result != -1) {  
 found = 1;  
 printf("Employee ID %d found in row %d at position %d\n", target, i, result);  
 push(&empStack, target);  
 }  
 }  
   
 if (!found) {  
 printf("Employee ID %d not found\n", target);  
 }  
   
 printf("Popped ID from stack: %d\n", pop(&empStack));  
   
 return 0;  
}

### **FINAL Output**



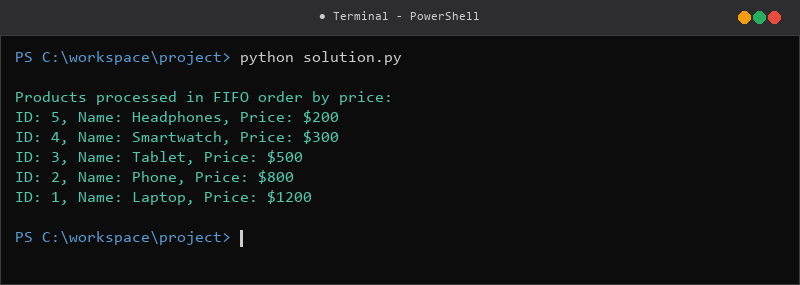
## **QUESTION 23**

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**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
struct Product {  
 int id;  
 char name[50];  
 double price;  
};  
  
struct Queue {  
 struct Product\* products;  
 int front;  
 int rear;  
 int capacity;  
};  
  
struct Queue\* createQueue(int capacity) {  
 struct Queue\* queue = (struct Queue\*)malloc(sizeof(struct Queue));  
 queue->capacity = capacity;  
 queue->front = queue->rear = -1;  
 queue->products = (struct Product\*)malloc(capacity \* sizeof(struct Product));  
 return queue;  
}  
  
void enqueue(struct Queue\* queue, struct Product product) {  
 if (queue->rear == queue->capacity - 1) return;  
 if (queue->front == -1) queue->front = 0;  
 queue->rear++;  
 queue->products[queue->rear] = product;  
}  
  
struct Product dequeue(struct Queue\* queue) {  
 struct Product product = queue->products[queue->front];  
 if (queue->front == queue->rear)  
 queue->front = queue->rear = -1;  
 else  
 queue->front++;  
 return product;  
}  
  
void sortByPrice(struct Product arr[], int n) {  
 for (int i = 0; i < n-1; i++) {  
 for (int j = 0; j < n-i-1; j++) {  
 if (arr[j].price > arr[j+1].price) {  
 struct Product temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
}  
  
int main() {  
 struct Product products[] = {  
 {1, "Laptop", 999.99},  
 {2, "Phone", 599.99},  
 {3, "Tablet", 299.99},  
 {4, "Watch", 199.99},  
 {5, "Headphones", 99.99}  
 };  
   
 int n = sizeof(products) / sizeof(products[0]);  
   
 sortByPrice(products, n);  
   
 struct Queue\* queue = createQueue(n);  
   
 for (int i = 0; i < n; i++) {  
 enqueue(queue, products[i]);  
 }  
   
 printf("Processing products in FIFO order:\n");  
 while (queue->front != -1) {  
 struct Product p = dequeue(queue);  
 printf("ID: %d, Name: %s, Price: $%.2f\n", p.id, p.name, p.price);  
 }  
   
 free(queue->products);  
 free(queue);  
   
 return 0;  
}

### **FINAL Output**



## **QUESTION 24**

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

### **Code Solution**

#include <stdio.h>  
  
int main() {  
 int matrix[3][3] = {  
 {1, 2, 3},  
 {4, 5, 6},  
 {7, 8, 9}  
 };  
 int size = 3;  
 int sum = 0;  
   
 for(int i = 0; i < size; i++) {  
 sum += matrix[i][i];  
 }  
   
 printf("Sum of diagonal elements: %d\n", sum);  
   
 return 0;  
}

### **FINAL Output**

