ASSIGNMENT 9

ASSIGNMENT 9

Q1 OUTPUT

```
BFS: 0 1 2 3
DFS: 0 1 2 3
PS C:\Users\tabhi\Desktop\dsa\graph>
```

Q2 Output

```
PS C:\Users\tabhi\Desktop\dsa\graph> cd
                                    > cd "c:\Users\tabhi\Desktop\dsa\graph\" ; if ($?) { g++ q2.cpp -o q2
} ; if ($?) { .\q2 }

Dijkstra:
Shortest distances from 0:
Node 0 = 0
Node 1 = 3
Node 2 = 1
Node 3 = 4


Kruskal MST:
Kruskal MST edges:
0 -- 2  (weight 1)
1 -- 3  (weight 1)
2 -- 1  (weight 2)
Total MST Weight = 4


Prim MST:
Prim MST edges:
2 -- 1  (weight 2)
0 -- 2  (weight 1)
1 -- 3  (weight 1)
Total MST Weight = 4
PS C:\Users\tabhi\Desktop\dsa\graph>
```

ASSIGNMENT 8

Q1

```
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 8> cd "c:\Users\tabhi\Desktop\
e2025\Assignment 8\" ; if ($?) { g++ Q1.cpp -o Q1 } ; if ($?) { .\Q1 }
Preorder: 1 2 3 4 5
Inorder: 2 1 4 3 5
Postorder: 2 4 5 3 1
Level Order: 1 2 3 4 5
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 8>
```

Q2

```
                                                    > cd "c:\Users\tab
e2025\Assignment 8\" ; if ($?) { g++ Q2.cpp -o Q2 } ; if ($?) { .\Q2 }
Predecessor: 4
Successor: 6
Search 7: Found
maximum: 9
minimum: 1
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 8>
```

Q3

```
                                                    > cd "c:\Users\tabhi\Desktop\dsa\Datastructur
e2025\Assignment 8\" ; if ($?) { g++ Q3.cpp -o Q3 } ; if ($?) { .\Q3 }
Inorder before deletion: 1 2 3 4 5 6
Inorder after deletion: 1 2 3 4 5
Max depth of tree: 3
Min depth of tree: 3
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 8>
```

Q4

```
                                                    > cd "c:\Users\tabhi\Desktop\dsa\Datas
e2025\Assignment 8\" ; if ($?) { g++ Q4.cpp -o Q4 } ; if ($?) { .\Q4 }
NO, it is NOT a BST
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 8>
```

Q 5

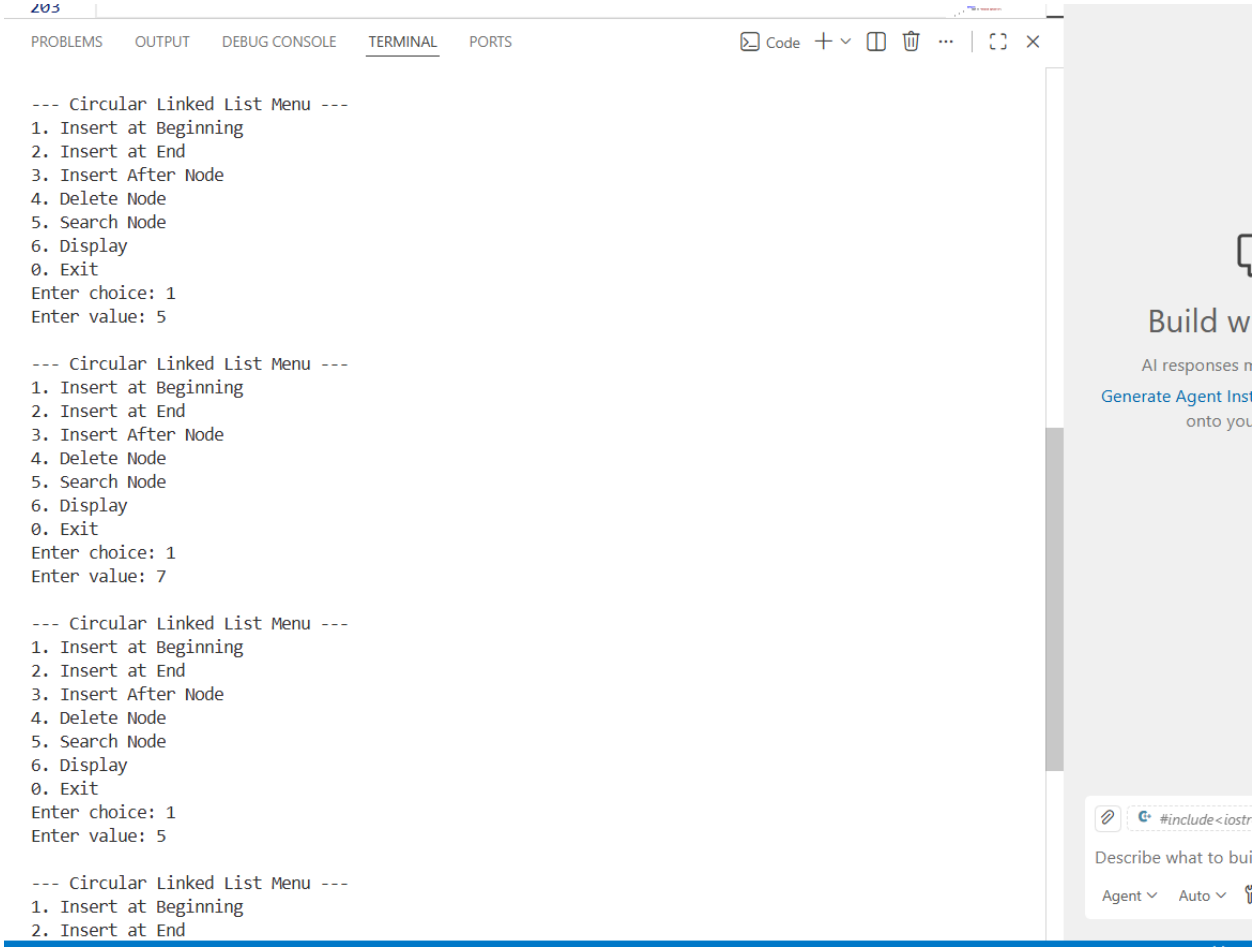
```
e2025\Assignment 8\" ; if ($?) { g++ Q5.cpp
Sorted Increasing Order: 1 3 6 8 10 15
Sorted Decreasing Order: 15 10 8 6 3 1
```

Q6

```
e2025\Assignment 8\" ; if ($?) { g++ Q6.cpp -o Q6 } ;
Top element: 60
After popping, new top: 50
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assigr
```

ASSIGNMENT 6

Q1 parta



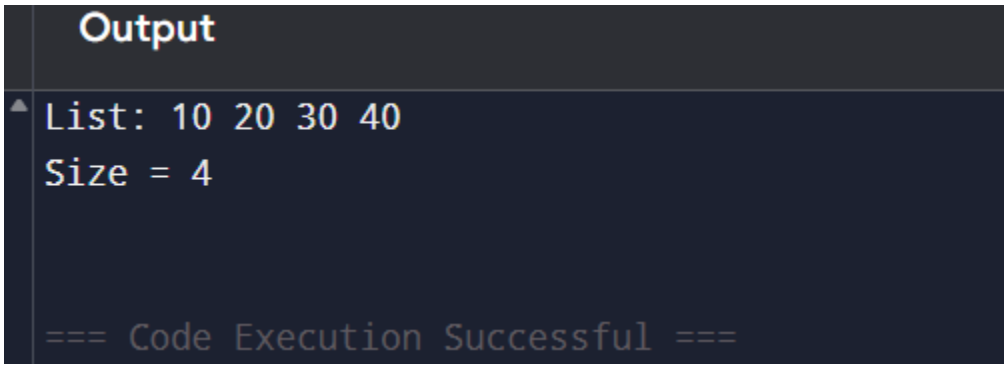
Q1 part b

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
e2025\Assignment 1\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCod
eRunnerFile }

--- Doubly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert After a Node
4. Insert Before a Node
5. Delete a Node
6. Search a Node
7. Display Forward
8. Display Backward
0. Exit
Enter your choice: 1
Enter value: 2

--- Doubly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert After a Node
4. Insert Before a Node
5. Delete a Node
6. Search a Node
7. Display Forward
8. Display Backward
0. Exit
Enter your choice: 2
Enter value: 1

--- Doubly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert After a Node
4. Insert Before a Node
5. Delete a Node
6. Search a Node
7. Display Forward
```

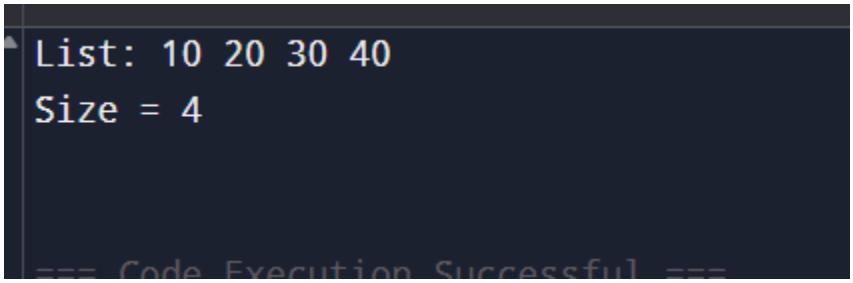
Q2

```
Output

Circular List Elements: 10  20  30  40  10

=== Code Execution Successful ===
```

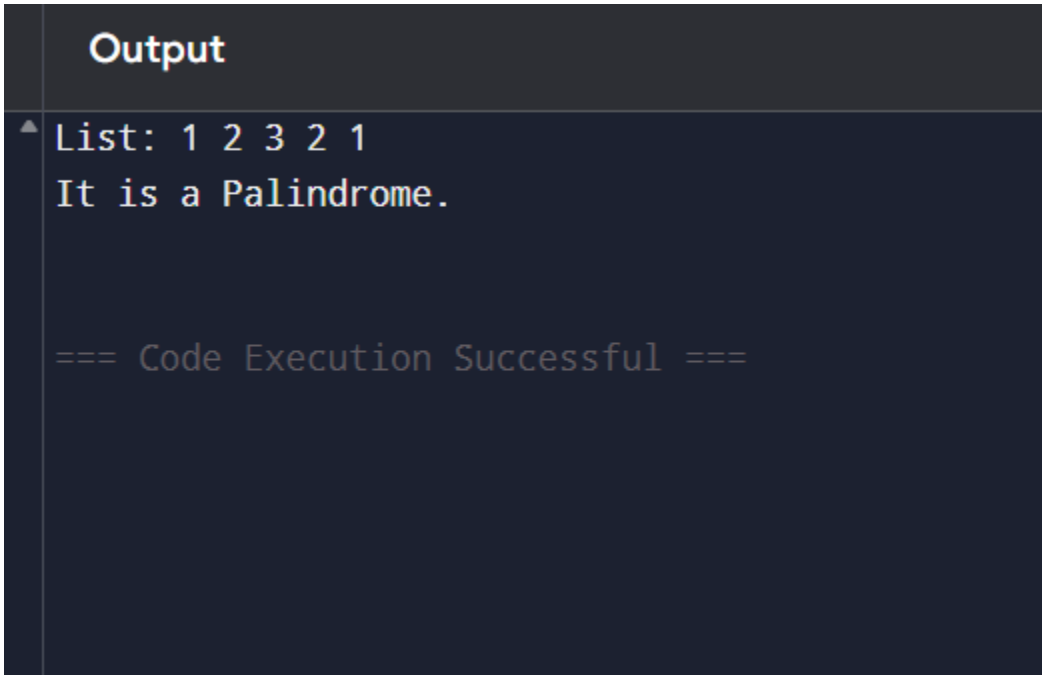
Q3part a



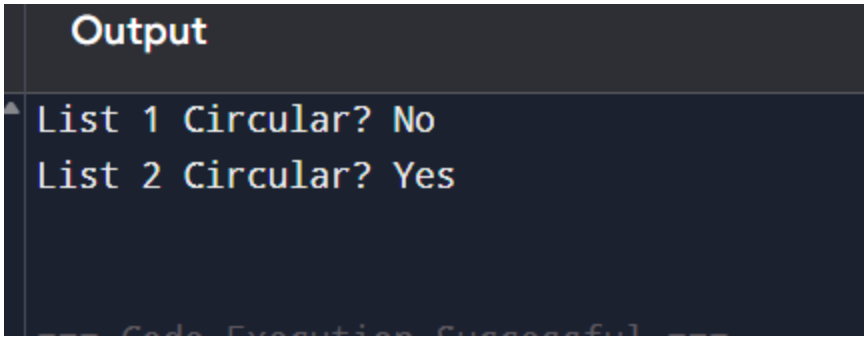
Q3 part b



Q4

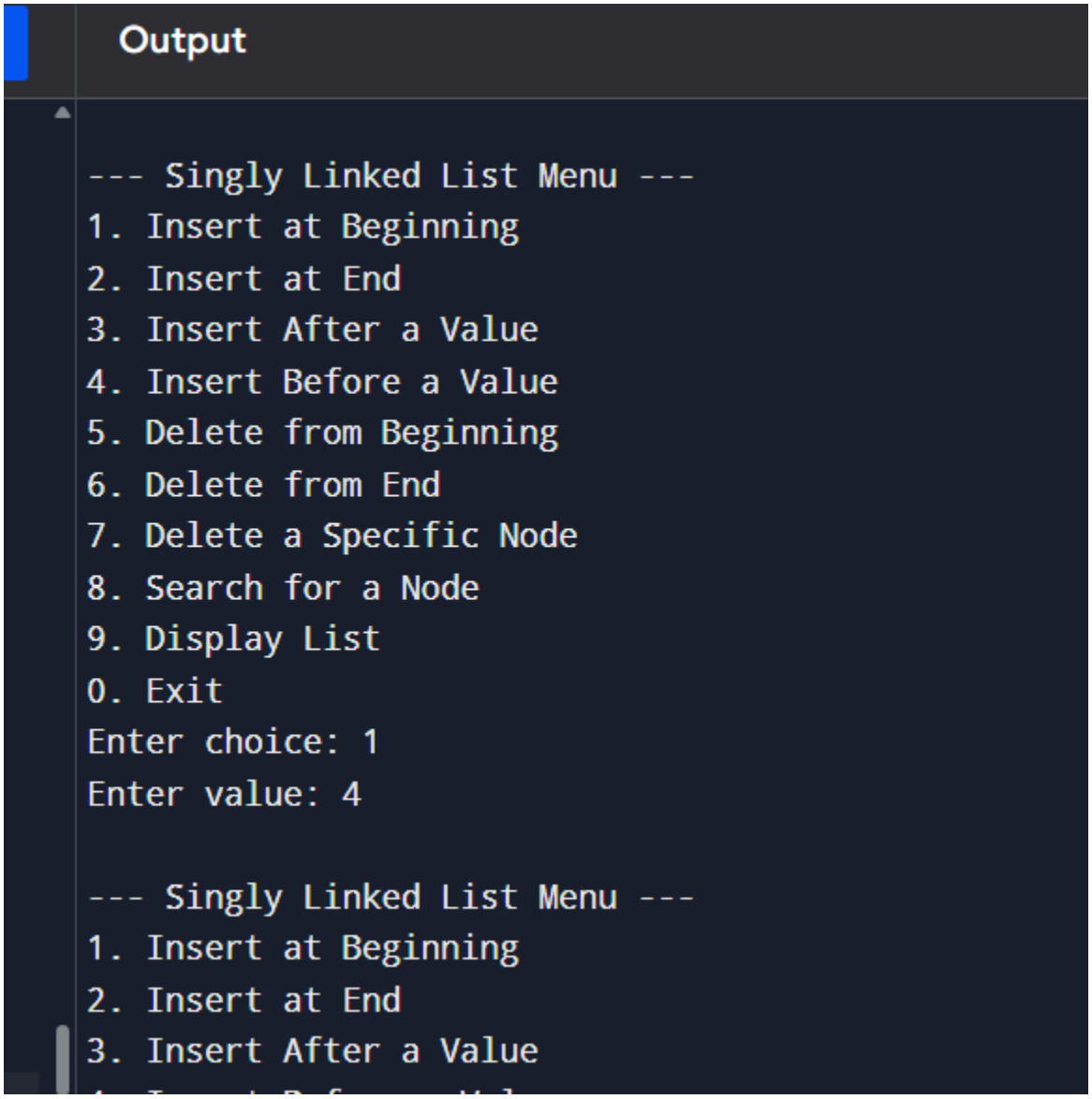


Q5


Output
List 1 Circular? No
List 2 Circular? Yes

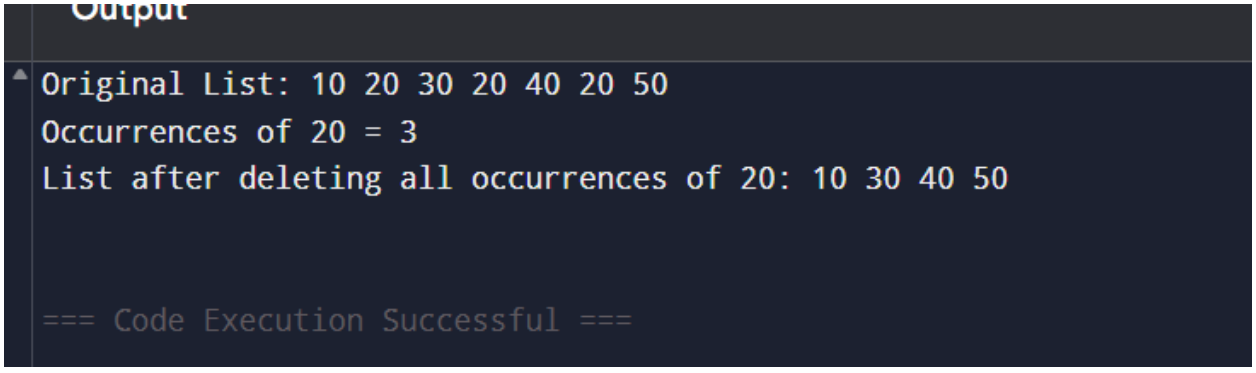
ASSIGNMENT 5

Q1



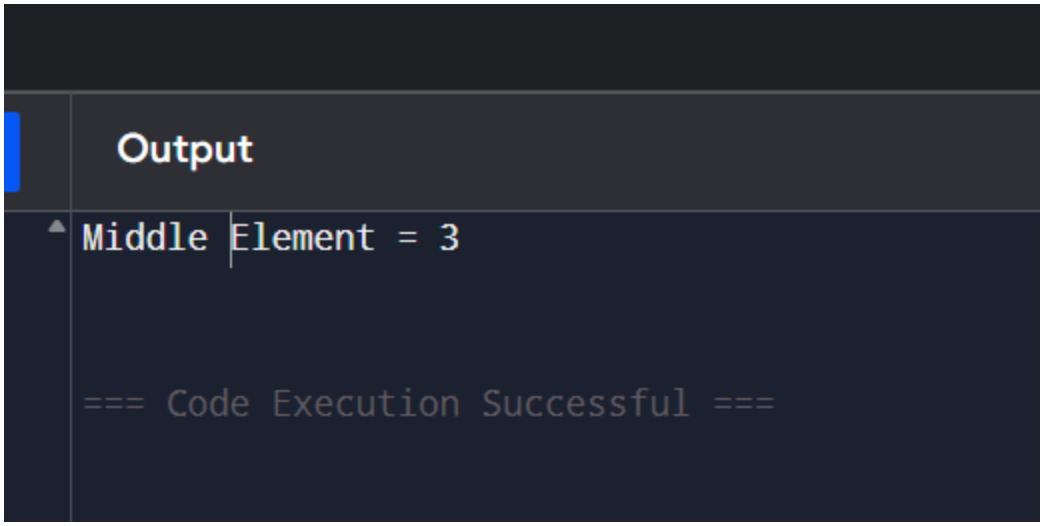
```
Output

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert After a Value
4. Insert Before a Value
5. Delete from Beginning
6. Delete from End
7. Delete a Specific Node
8. Search for a Node
9. Display List
0. Exit
Enter choice: 1
Enter value: 4

--- Singly Linked List Menu ---
1. Insert at Beginning
2. Insert at End
3. Insert After a Value
```

Q2

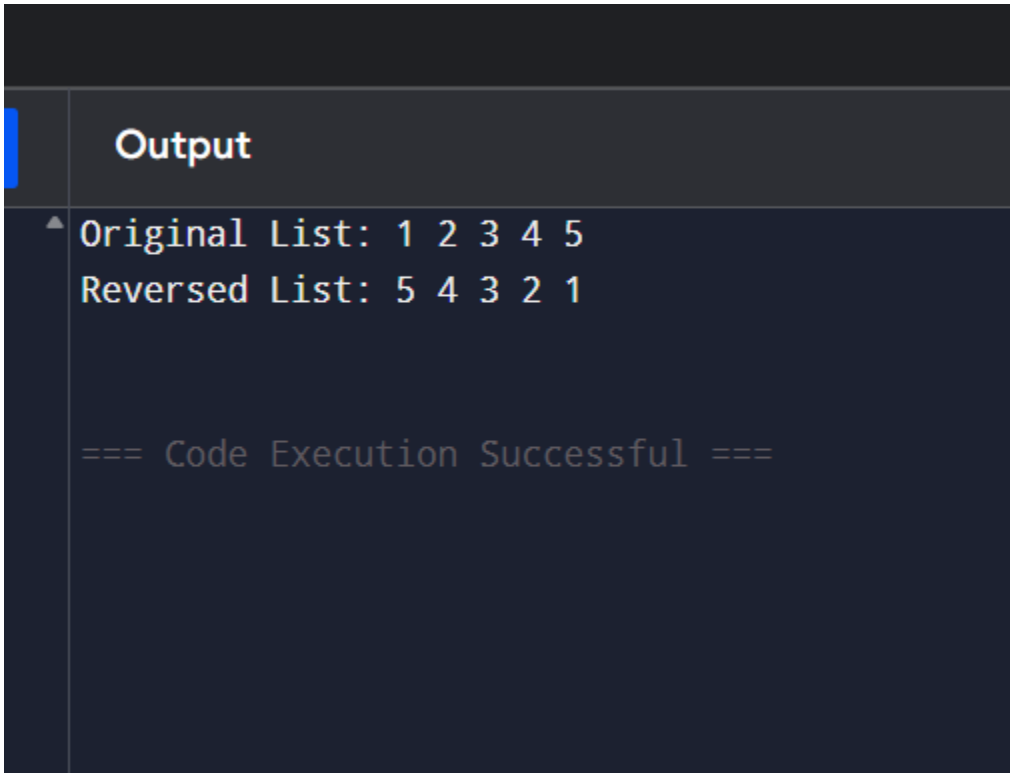


```
Output

Original List: 10 20 30 20 40 20 50
Occurrences of 20 = 3
List after deleting all occurrences of 20: 10 30 40 50


=== Code Execution Successful ===
```

Q3


Output
Middle Element = 3
=== Code Execution Successful ===

Q4


Output
Original List: 1 2 3 4 5
Reversed List: 5 4 3 2 1
=== Code Execution Successful ===

ASSIGNMENT 4

Output

```
1.Enqueue
2.Dequeue
3.Peek
4.Display
5.isEmpty
6.isFull
7.Size
8.Capacity
0.Exit
1
24

1.Enqueue
2.Dequeue
3.Peek
4.Display
5.isEmpty
6.isFull
7.Size
```

Q1

Q2

Output

```
1.Enqueue
2.Dequeue
3.Peek
4.Display
5.isEmpty
6.isFull
7.Size
8.Capacity
0.Exit
1
34

1.Enqueue
2.Dequeue
3.Peek
4.Display
5.isEmpty
6.isFull
7.Size
```

Q3

```
Output

Original queue: 4 7 11 20 5 9
Interleaved queue: 4 20 7 5 11 9


=== Code Execution Successful ===
```

Q4

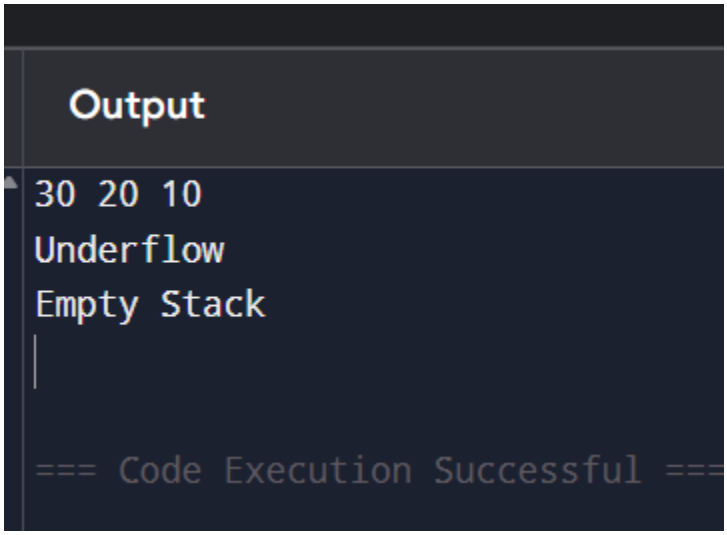
```
Output

Input stream: aabc
First Non-Repeating Characters: a -1 b b


=== Code Execution Successful ===
```

Q5

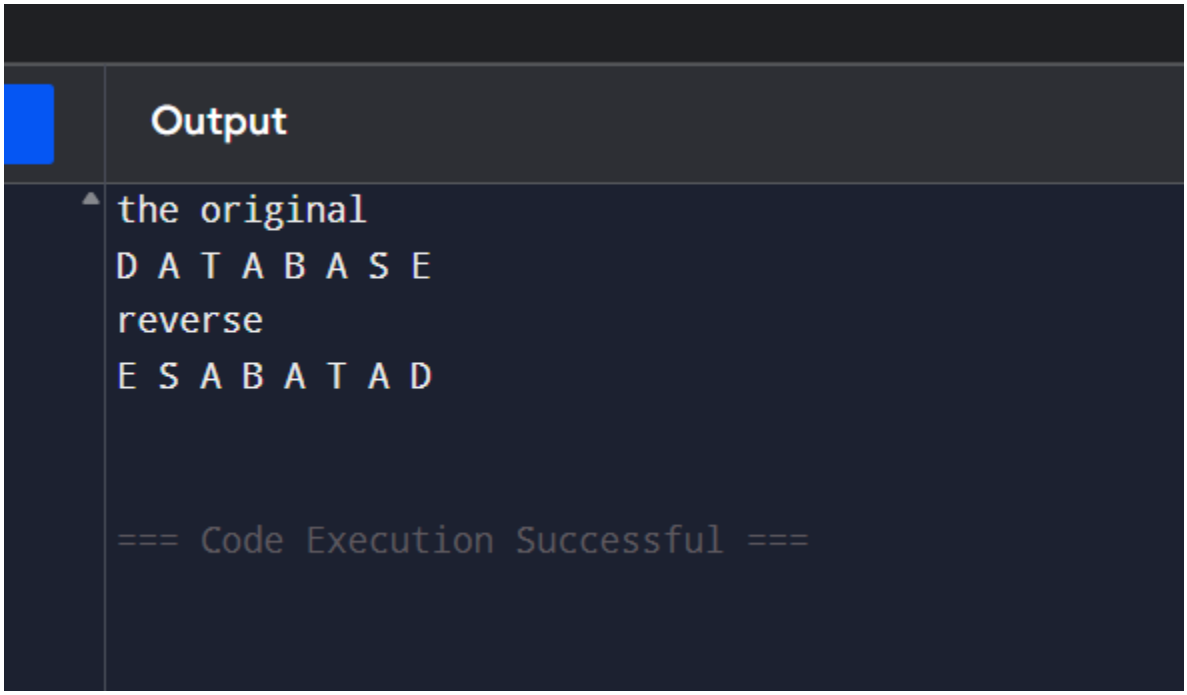
```
Top (two queues): 30
Top (one queue): 25


=== Code Execution Successful ===
```

ASSIGNMENT 3

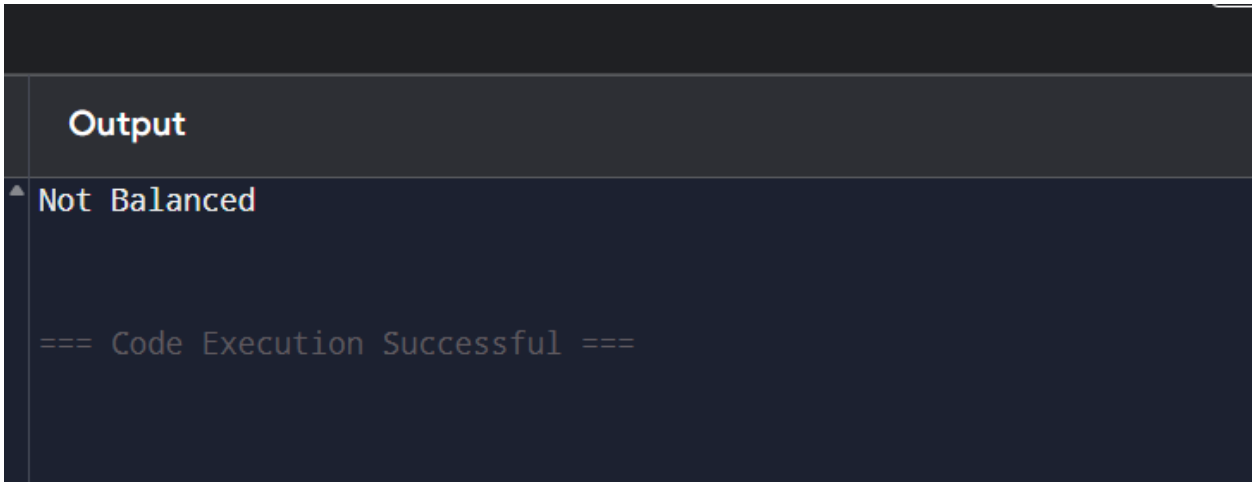
Q1


Output
30 20 10
Underflow
Empty Stack
=== Code Execution Successful ===

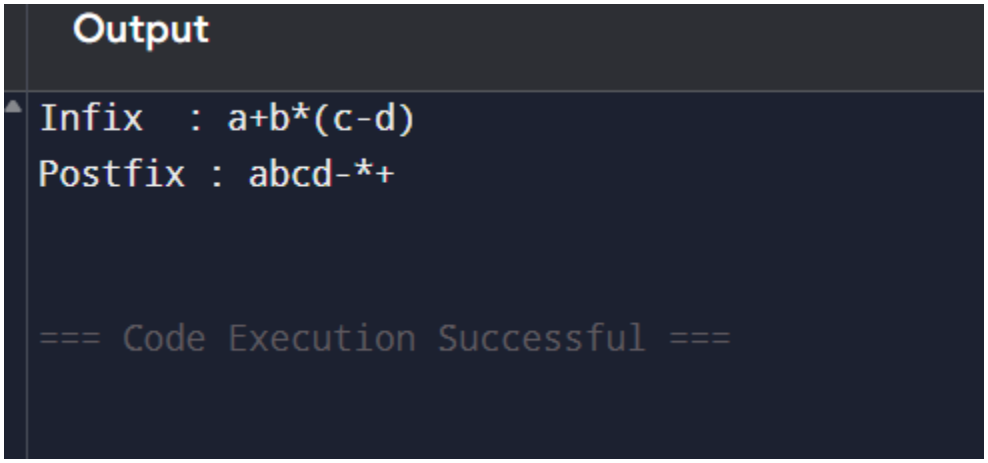
Q2


Output
the original
D A T A B A S E
reverse
E S A B A T A D
=== Code Execution Successful ===

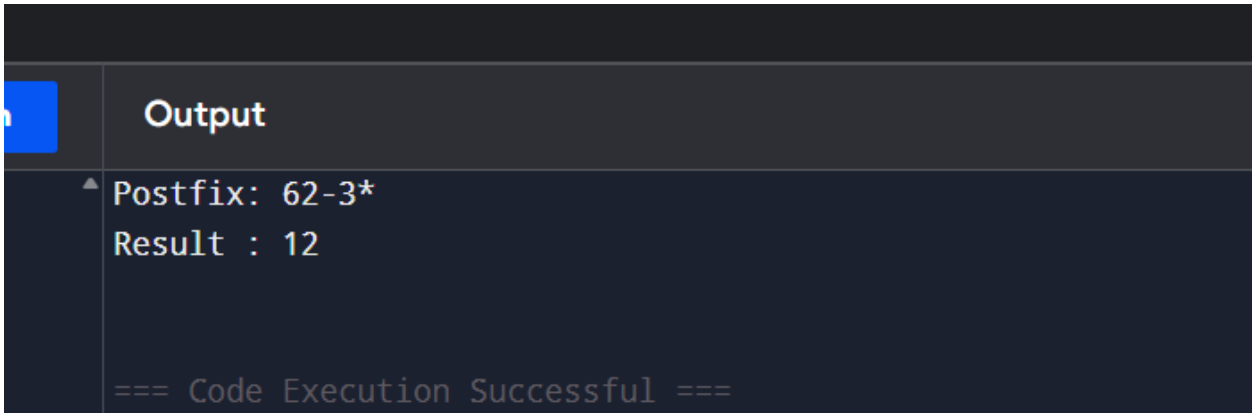
Q3

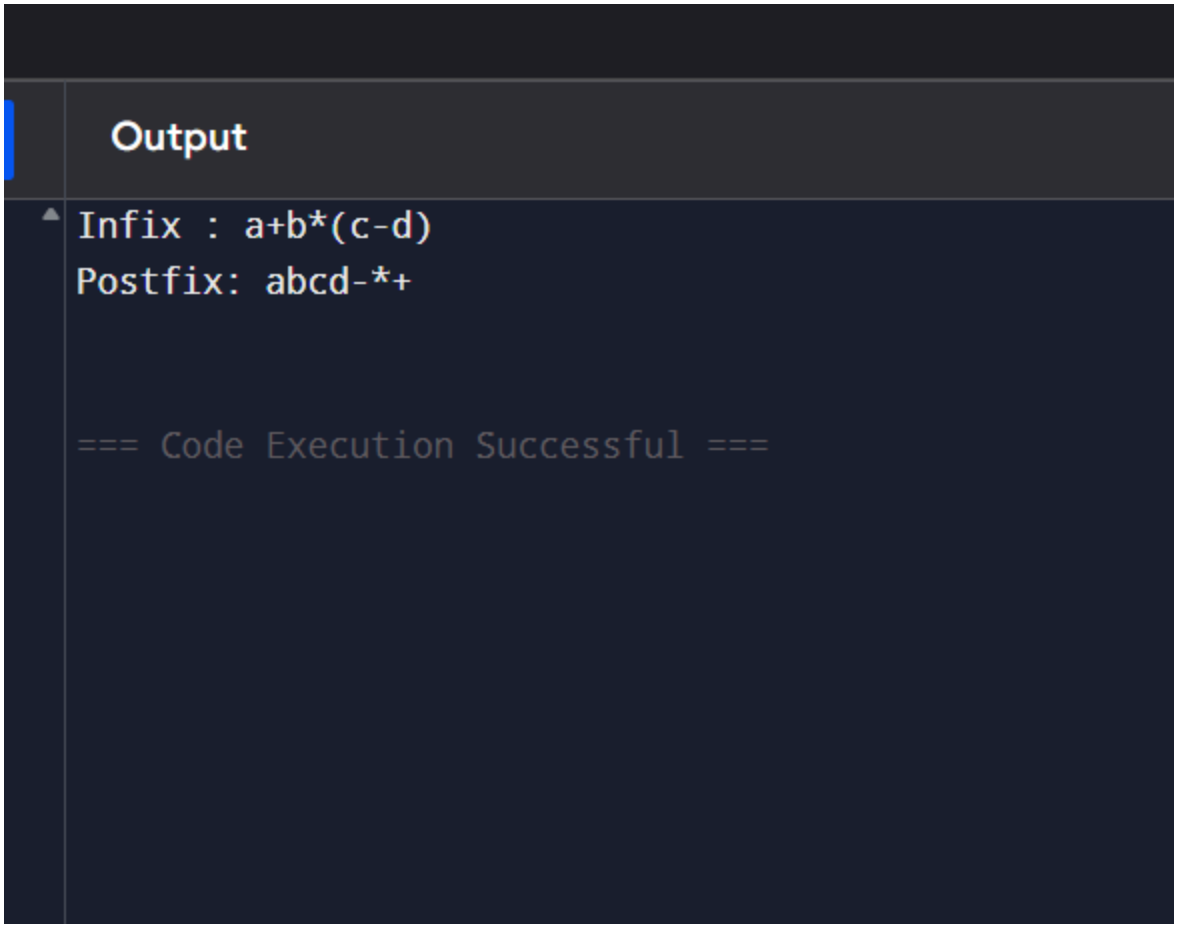

Output
Not Balanced
=== Code Execution Successful ===

Q4


Output
Infix : a+b*(c-d)
Postfix : abcd-*+
=== Code Execution Successful ===

Q5

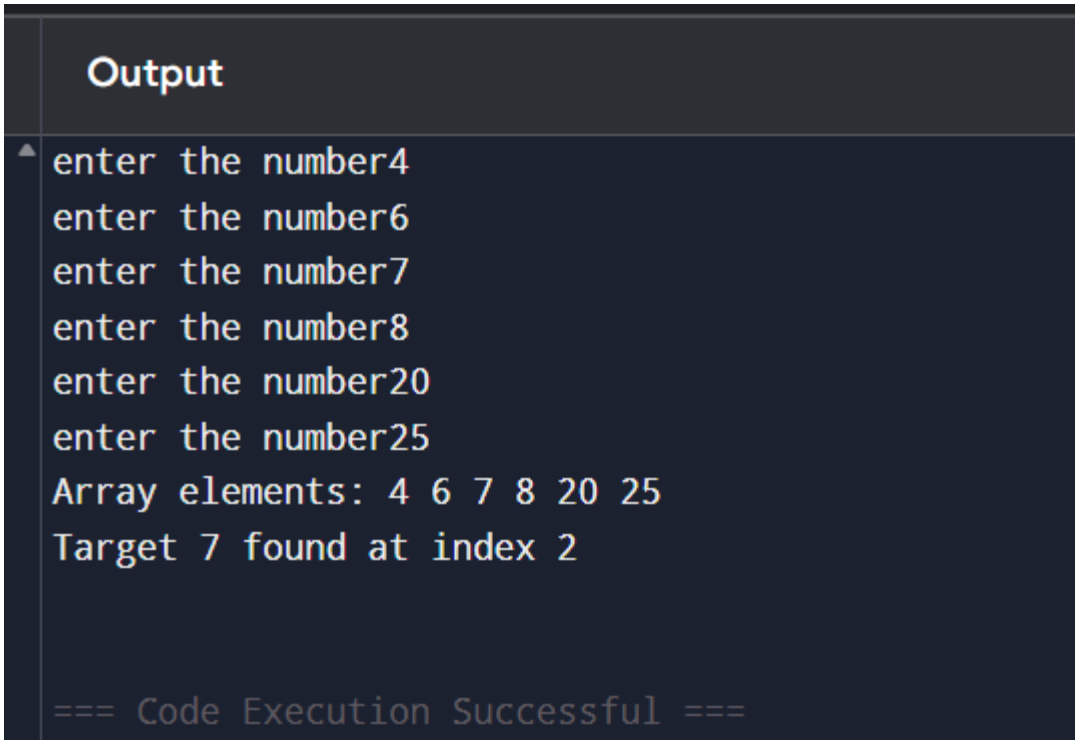

Output
Postfix: 62-3*
Result : 12
=== Code Execution Successful ===



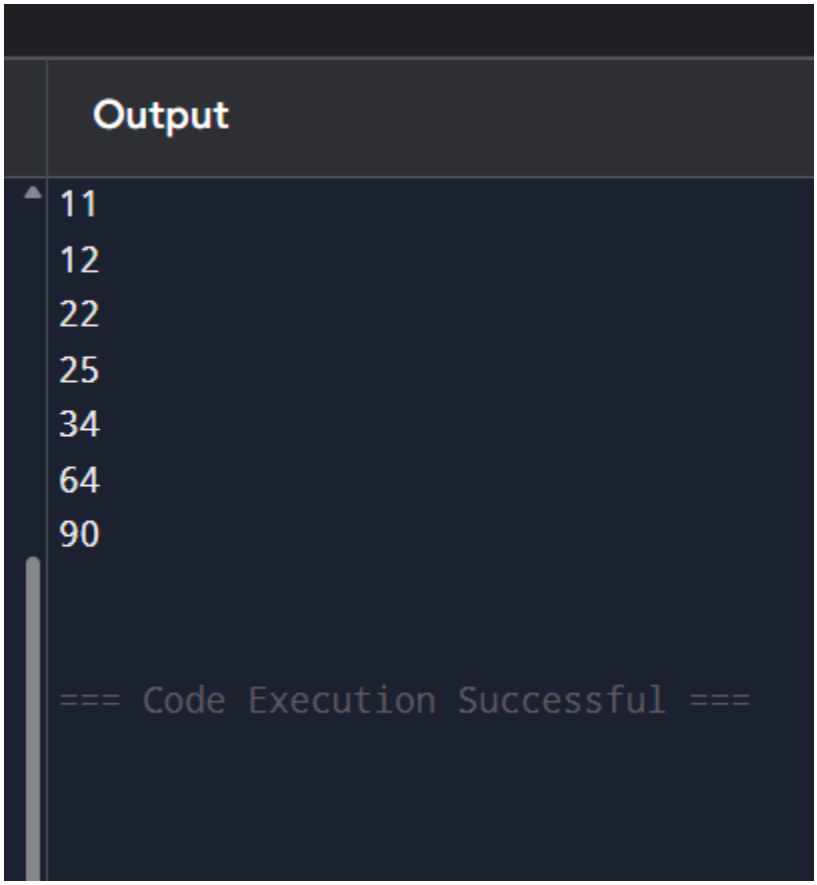
Q6’
Q7

ASSIGNMENT 2

Q1


Output
enter the number4
enter the number6
enter the number7
enter the number8
enter the number20
enter the number25
Array elements: 4 6 7 8 20 25
Target 7 found at index 2
=== Code Execution Successful ===

Q2


Output
11
12
22
25
34
64
90
=== Code Execution Successful ===

Q3

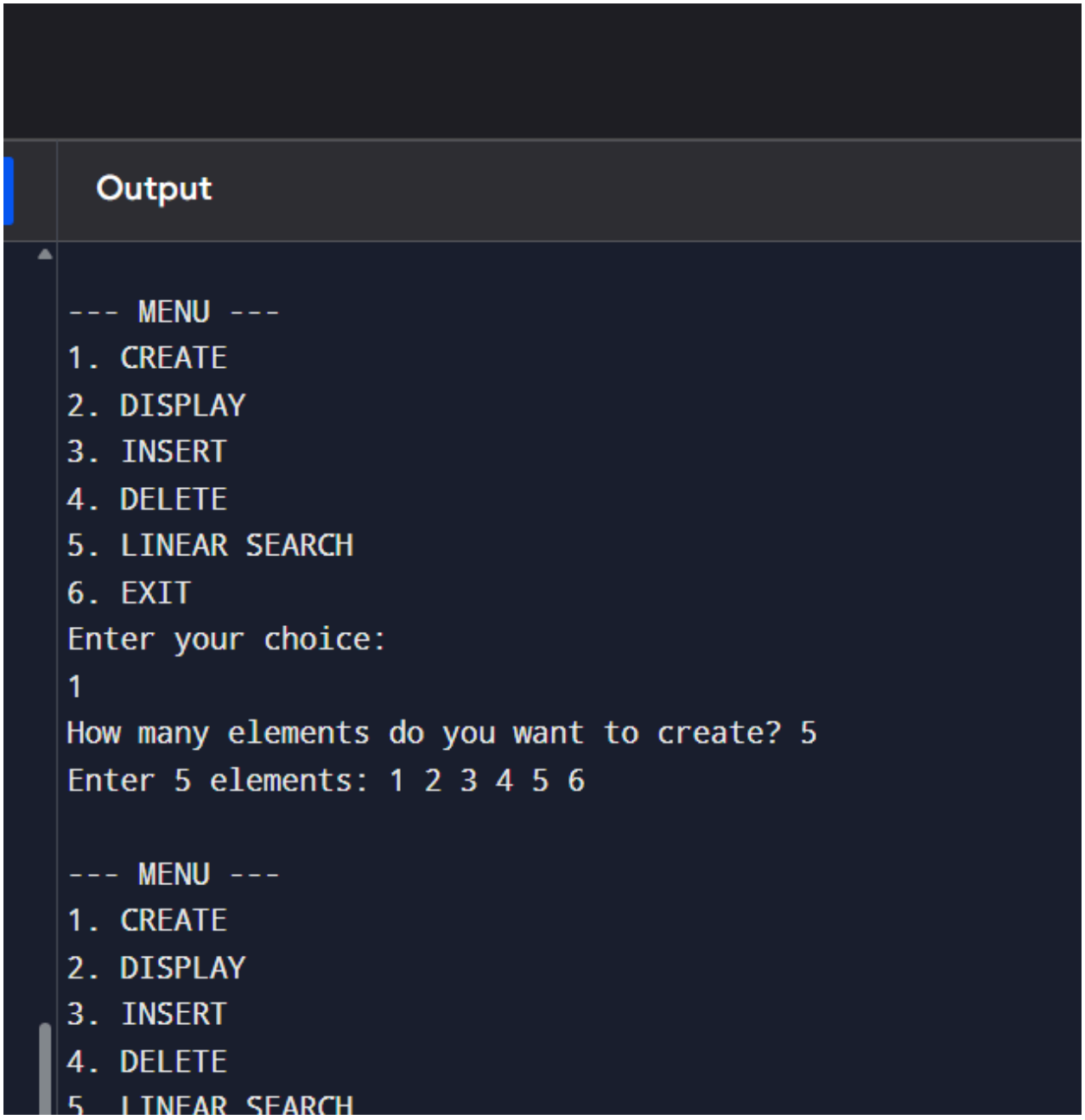
```
Output

the missing element is by using linear search5
Missing number (Binary Search): 5


=== Code Execution Successful ===
```

Q4PART A

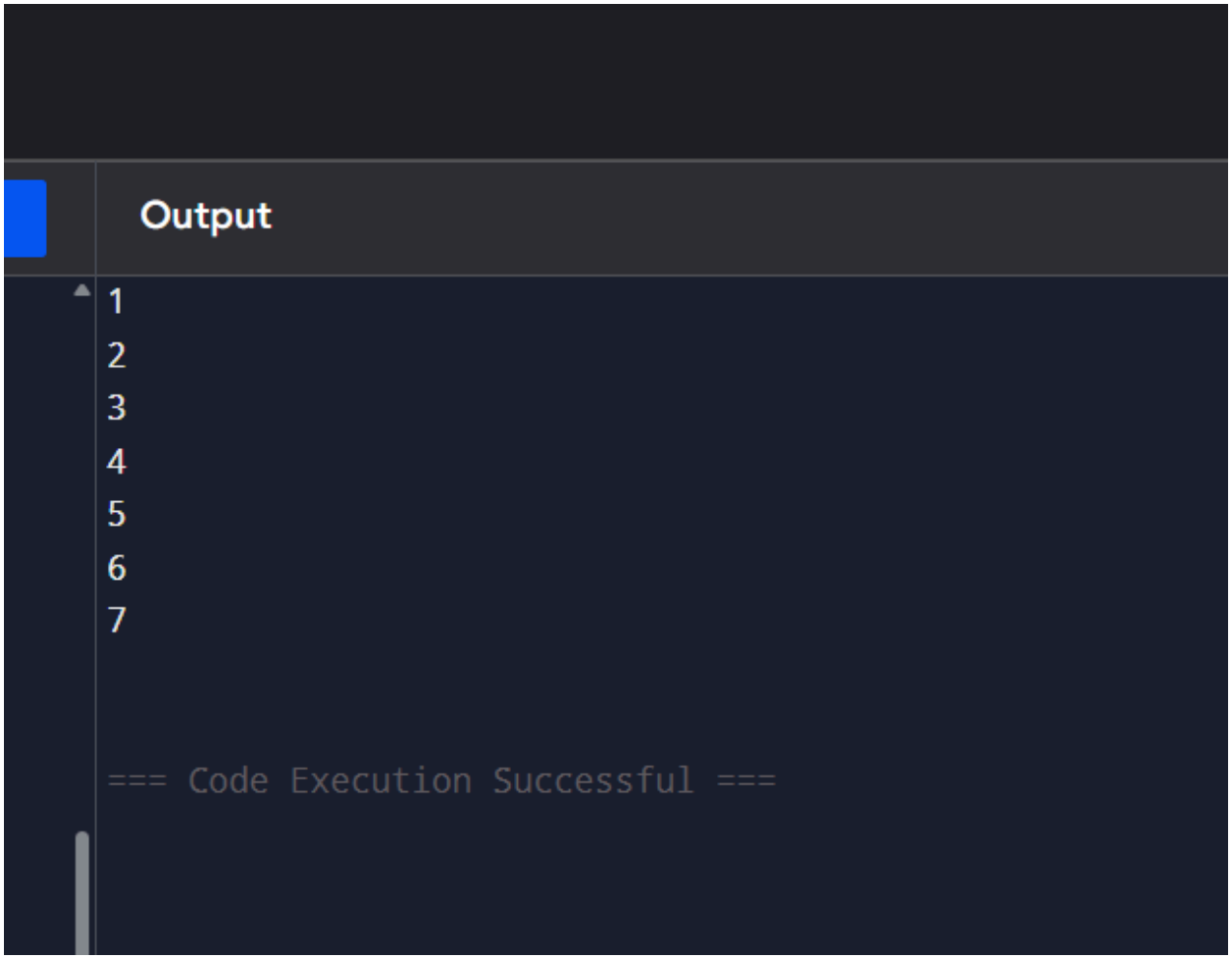
```
2025\Assignment 2\" ; if ($?) { g++ Q4part1.cpp -o Q4part1 } ; if ($?) { .\Q4part
enter the first string Abhinav
enter the second string Gupta
the concatinated strings are Abhinav Gupta
PS C:\Users\tabhi\Desktop\dsa\Datastructure2025\Assignment 2>
```

ASSIGNMENT 1

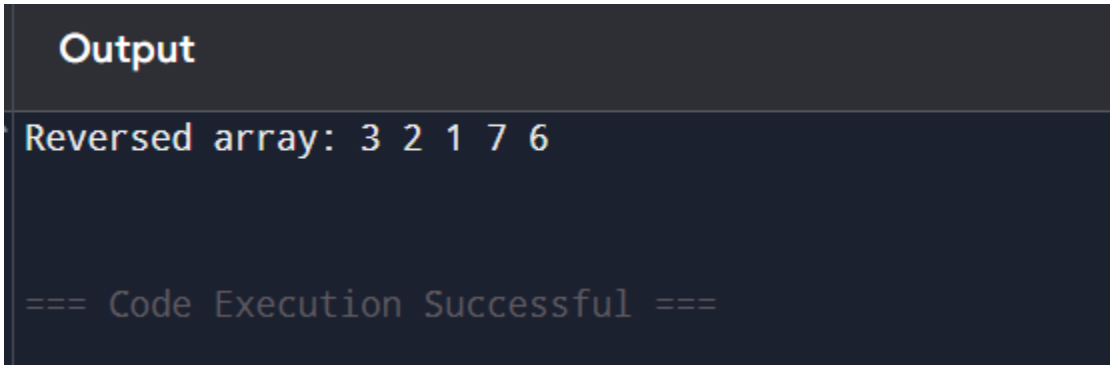
Q1


Output
--- MENU ---
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. LINEAR SEARCH
6. EXIT
Enter your choice:
1
How many elements do you want to create? 5
Enter 5 elements: 1 2 3 4 5 6
--- MENU ---
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. LINEAR SEARCH

Q2


Output
1
2
3
4
5
6
7
=== Code Execution Successful ===

Q4(a)


Output
Reversed array: 3 2 1 7 6
=== Code Execution Successful ===

```
Output

Matrix multiplication result:
58 64
139 154


=== Code Execution Successful ===
```

Q4 (b)

Q4 (c)

```
Output

Transpose of the matrix:
1 4
2 5
3 6


=== Code Execution Successful ===
```

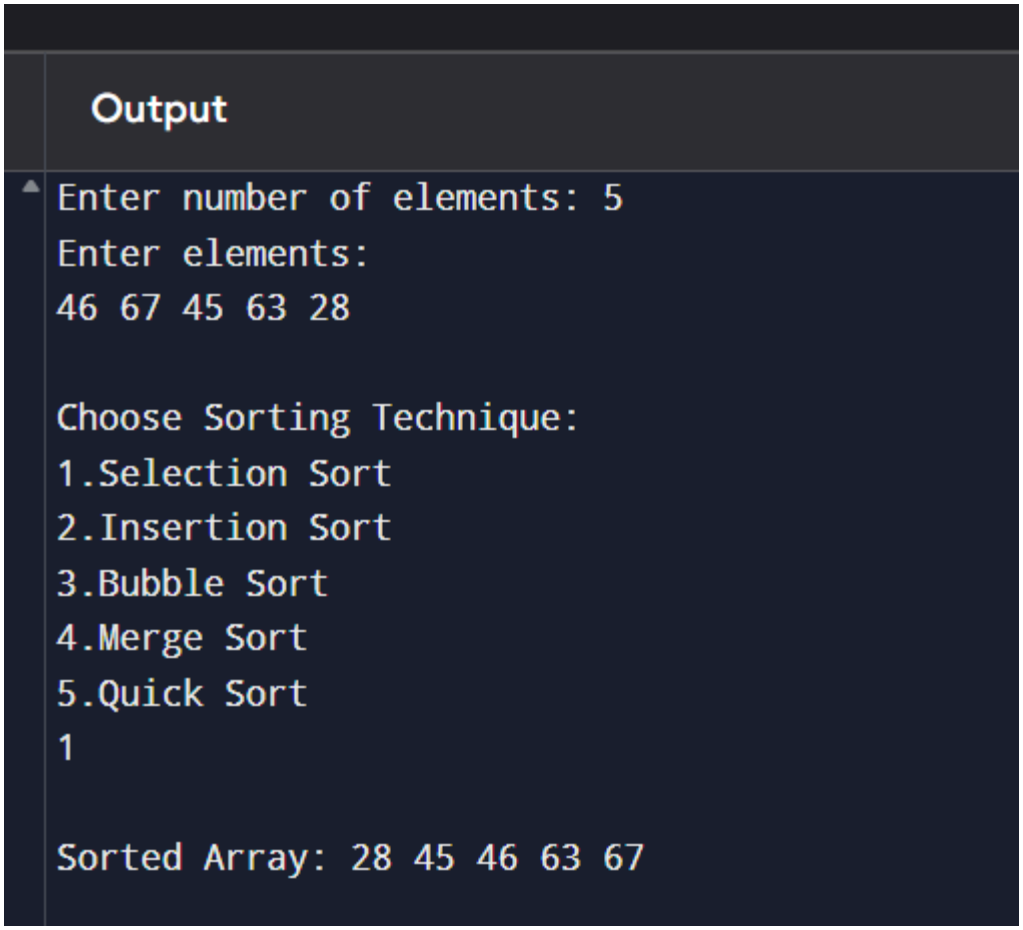
Q5

Output

```
Hey! Enter number of rows: 2
Cool, now enter number of columns: 3
Enter the elements row by row:
1 2 3 4 5 6
Sum of row 1 is 6
Sum of row 2 is 15
Sum of column 1 is 5
Sum of column 2 is 7
Sum of column 3 is 9
Done! All sums calculated successfully.


=== Code Execution Successful ===
```

Assignment 7

Q1


Output
Enter number of elements: 5
Enter elements:
46 67 45 63 28
Choose Sorting Technique:
1.Selection Sort
2.Insertion Sort
3.Bubble Sort
4.Merge Sort
5.Quick Sort
1
Sorted Array: 28 45 46 63 67