LAB TASK

LAB TASK NO 6

SUBJECT: DATA STRUCTURE AND ALGORITHM

SUBMITTED TO: HIJAB DURRANI

SUBMITTED BY: MISBAH ULLAH JAN

CLASS CODE: BSSE-2024A

SEMESTER: 3RD

DATE: 11/27/2025

DEPARTMENT OF: SOFTWARE ENGINEERING

CECOS UNIVERSITY HAYATABAD PESHAWAR

LAB TASK 1:

1. Modify your stack program so that:

1. **Before inserting (enqueue), the program checks if the value already exists.**
2. **If it exists, do not insert and print:**

Duplicate value! Insertion not allowed.

Otherwise insert normally.

Algorithm + Modified C++ Program

Algorithm: Push with Duplicate Check

1. Start
2. Input value to push
3. IF stack is full
   → Print "Stack Overflow! Cannot push."
   → Stop
4. Traverse the stack from 0 to top
   - IF any element equals the new value
     → Print "Duplicate value! Insertion not allowed."
     → Stop
5. Increment top
6. Insert value at stackArr[top]
7. Print "{value} pushed successfully"
8. End

CODE:

main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;

#define SIZE 5
int stackArr[SIZE];
int top = -1;

// Check empty
bool isEmpty() {
    return top == -1;
}

// Check full
bool isFull() {
    return top == SIZE - 1;
}

// Check duplicate
bool isDuplicate(int value) {
    for (int i = 0; i <= top; i++) {
        if (stackArr[i] == value)
            return true;
    }
    return false;
}

// Push operation with duplicate check
void push(int value) {
```

main.cpp

Share Run

```cpp

// Push operation with duplicate check
void push(int value) {
    if (isFull()) {
        cout << "Stack Overflow! Cannot push.\n";
        return;
    }

    if (isDuplicate(value)) {
        cout << "Duplicate value! Insertion not allowed.\n";
        return;
    }

    top++;
    stackArr[top] = value;
    cout << value << " pushed into stack.\n";
}

// Pop operation
void pop() {
    if (isEmpty()) {
        cout << "Stack Underflow! Nothing to pop.\n";
        return;
    }
    cout << stackArr[top] << " popped from stack.\n";
    top--;
}

```

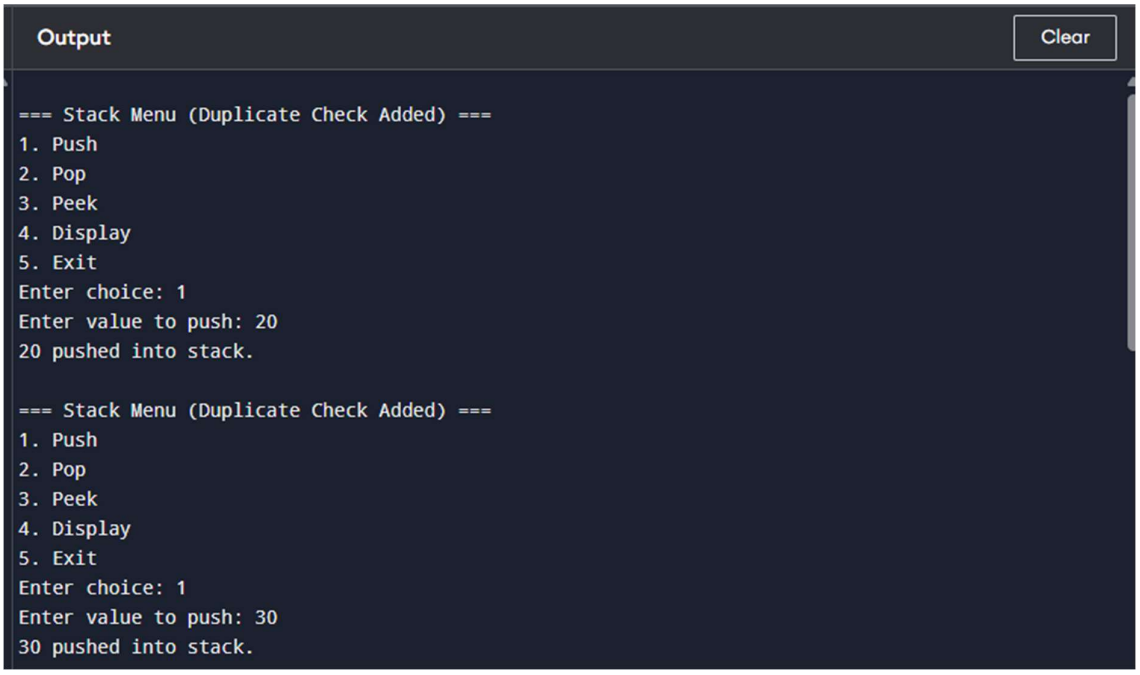
main.cpp

Share Run

```cpp
// Peek operation
void peek() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
    } else {
        cout << "Top element: " << stackArr[top] << endl;
    }
}

// Display elements
void display() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
        return;
    }
    cout << "Stack elements: ";
    for (int i = top; i >= 0; i--) {
        cout << stackArr[i] << " ";
    }
    cout << endl;
}

int main() {
    int choice, value;

    while (true) {
        cout << "\n=== Stack Menu (Duplicate Check Added) ===\n";
        cout << "1. Push\n";
        cout << "2. Pop\n";
        cout << "3. Peek\n";
        cout << "4. Display\n";
        cout << "5. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to push: ";
                cin >> value;
                push(value);
                break;

            case 2:
                pop();
                break;

            case 3:
                peek();
                break;

            case 4:
                display();
                break;
```

```
            case 5:
                cout << "Exiting...\n";
                return 0;

            default:
                cout << "Invalid choice! Try again.\n";
        }
    }
}
```

CODE OUTPUT:

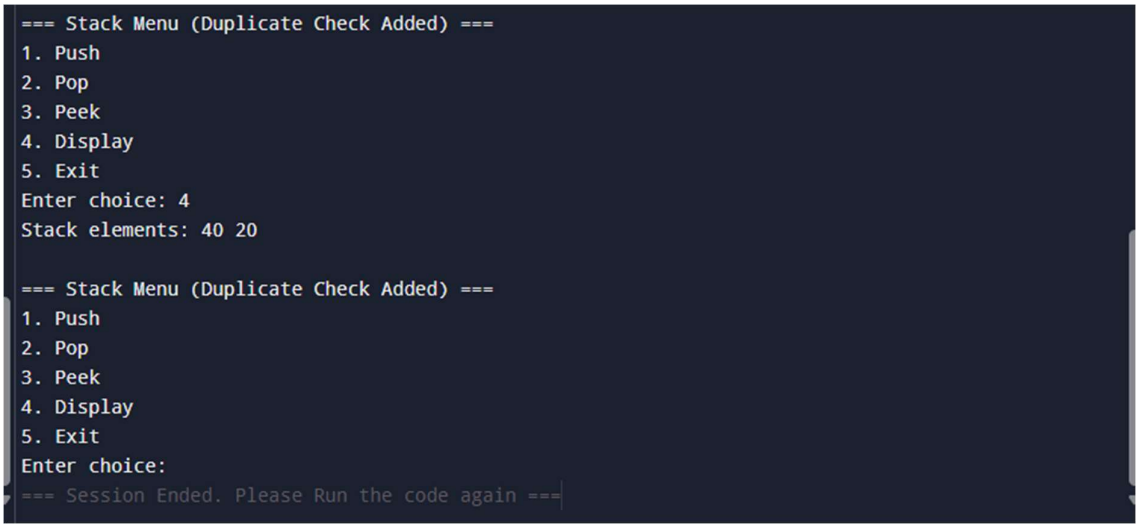


Output Clear

```
=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 20
Duplicate value! Insertion not allowed.

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 4
Stack elements: 30 20

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 2
30 popped from stack.
```

Output Clear

```
=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 20
Duplicate value! Insertion not allowed.

=== Stack Menu (Duplicate Check Added) ===
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 40
40 pushed into stack.
```



LAB TASK 2:

2. Write a program using a stack (array) that adds an additional function:

countElements()

This function returns the number of elements currently in the queue.

Example:

If queue contains: [10, 20, 30]

countElements() → 3

Algorithm for countElements()

Algorithm: countElements()

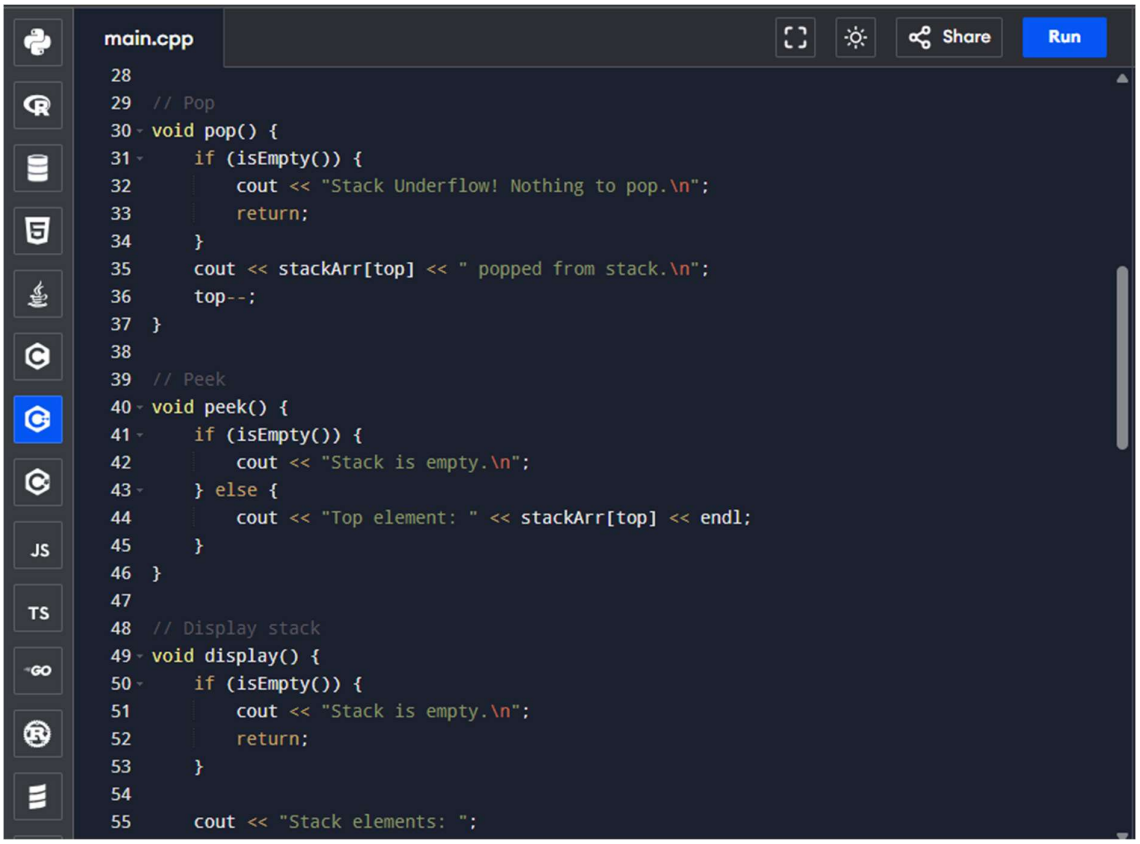
1. Start
2. If stack is empty (top == -1):
   → Return 0
3. Otherwise, number of elements = top + 1
4. Return top + 1
5. End

CODE :

main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;

#define SIZE 5
int stackArr[SIZE];
int top = -1;

// Check empty
bool isEmpty() {
    return top == -1;
}

// Check full
bool isFull() {
    return top == SIZE - 1;
}

// Push
void push(int value) {
    if (isFull()) {
        cout << "Stack Overflow! Cannot push.\n";
        return;
    }
    top++;
    stackArr[top] = value;
    cout << value << " pushed into stack.\n";
}
```



main.cpp

Share Run

```cpp

// Pop
void pop() {
    if (isEmpty()) {
        cout << "Stack Underflow! Nothing to pop.\n";
        return;
    }
    cout << stackArr[top] << " popped from stack.\n";
    top--;
}

// Peek
void peek() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
    } else {
        cout << "Top element: " << stackArr[top] << endl;
    }
}

// Display stack
void display() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
        return;
    }

    cout << "Stack elements: ";
```

main.cpp

Share Run

```
    for (int i = top; i >= 0; i--) {
        cout << stackArr[i] << " ";
    }
    cout << endl;
}

// New function: Count elements in stack
int countElements() {
    return top + 1;  // because top starts at -1
}

int main() {
    int choice, value;

    while (true) {
        cout << "\n=== Stack Menu (With countElements) ===\n";
        cout << "1. Push\n";
        cout << "2. Pop\n";
        cout << "3. Peek\n";
        cout << "4. Display\n";
        cout << "5. Count Elements\n";
        cout << "6. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to push: ";
```

```
                cout << "Enter value to push: ";
                cin >> value;
                push(value);
                break;

            case 2:
                pop();
                break;

            case 3:
                peek();
                break;

            case 4:
                display();
                break;

            case 5:
                cout << "Total elements in stack = " << countElements() << endl;
                break;

            case 6:
                cout << "Exiting...\n";
                return 0;

            default:
                cout << "Invalid choice! Try again.\n";
        }
```

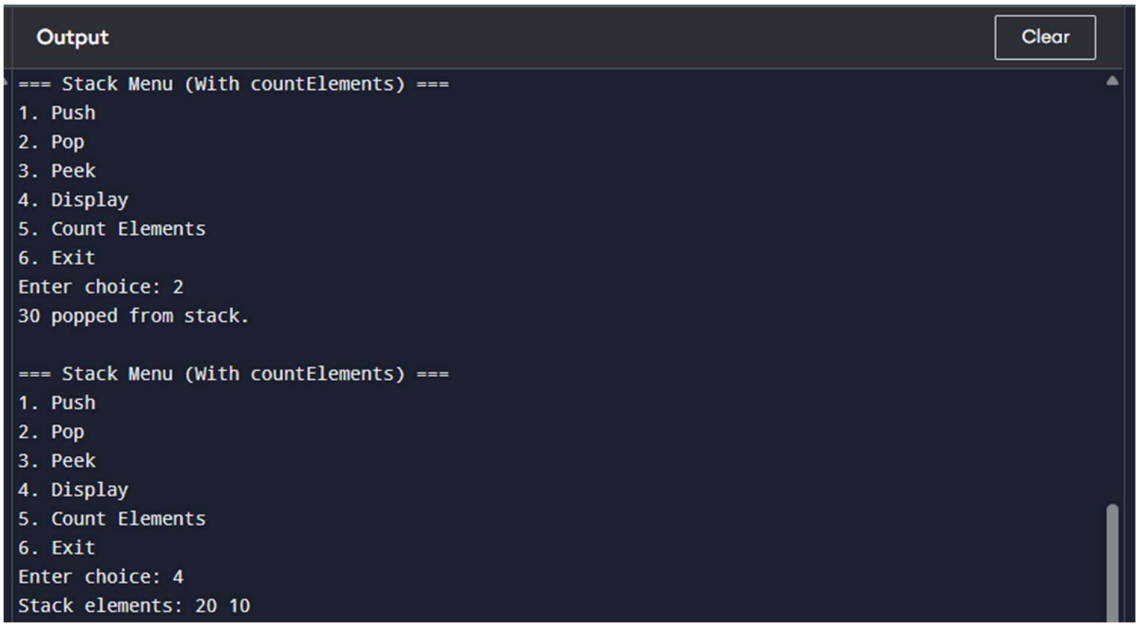
CODE OUTPUT:

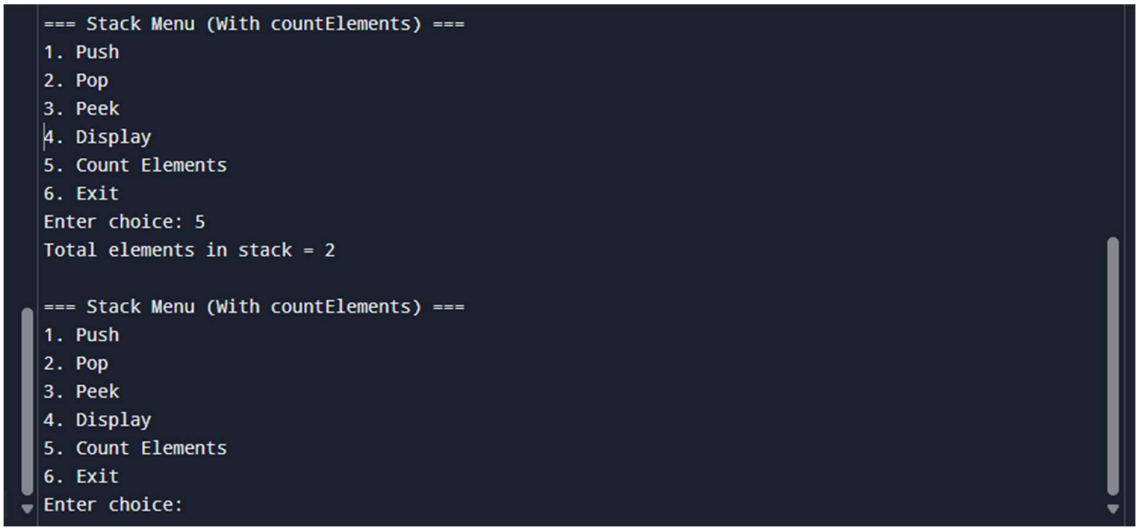
Output Clear

```
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 10
10 pushed into stack.

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 20
20 pushed into stack.
```

Output Clear

```
=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 1
Enter value to push: 30
30 pushed into stack.

=== Stack Menu (With countElements) ===
1. Push
2. Pop
3. Peek
4. Display
5. Count Elements
6. Exit
Enter choice: 4
Stack elements: 30 20 10
```





LAB TASK 3:

3. Write a program to:

1. Insert values into a queue using an array

2. Reverse the stack without using any library functions

3. Display the reversed stack

Example:

Input: 10 20 30 40

Output: 40 30 20 10

Algorithm:

Step 1 — Insert values into queue (array-based queue)

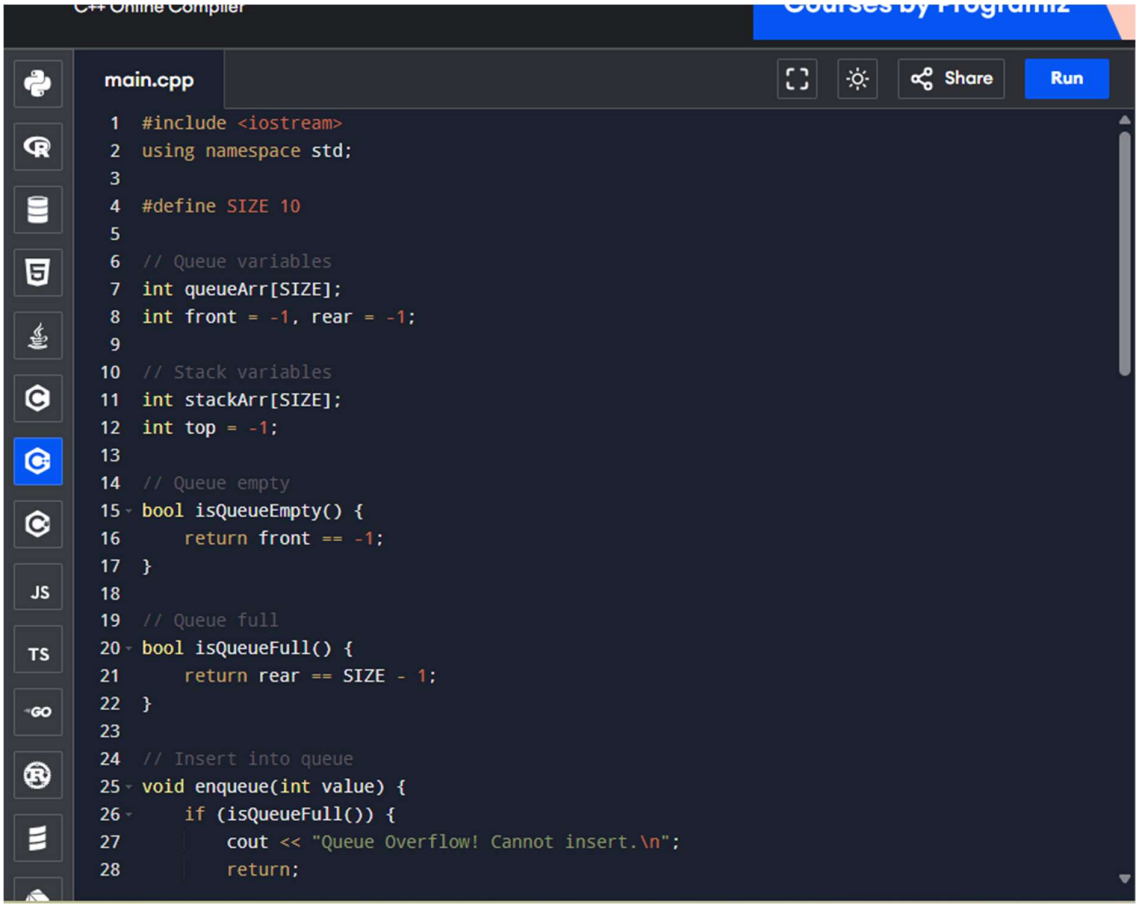
1. Start
2. For each input value:
    - If queue is full → print overflow
    - Else insert at rear and increment rear

Step 2 — Reverse using a stack (array)

1. Create an empty stack
2. While queue is not empty:
    - Pop each queue element and push it onto the stack
3. Now pop elements from stack to get reversed order

Step 3 — Display reversed stack

1. Print stack elements from top down to 0


C++ Online Compiler
Courses by Programiz
main.cpp
Share
Run
JS
TS
GO
#include <iostream>
using namespace std;
#define SIZE 10
// Queue variables
int queueArr[SIZE];
int front = -1, rear = -1;
// Stack variables
int stackArr[SIZE];
int top = -1;
// Queue empty
bool isQueueEmpty() {
return front == -1;
}
// Queue full
bool isQueueFull() {
return rear == SIZE - 1;
}
// Insert into queue
void enqueue(int value) {
if (isQueueFull()) {
cout << "Queue Overflow! Cannot insert.\n";
return;

main.cpp

Share Run

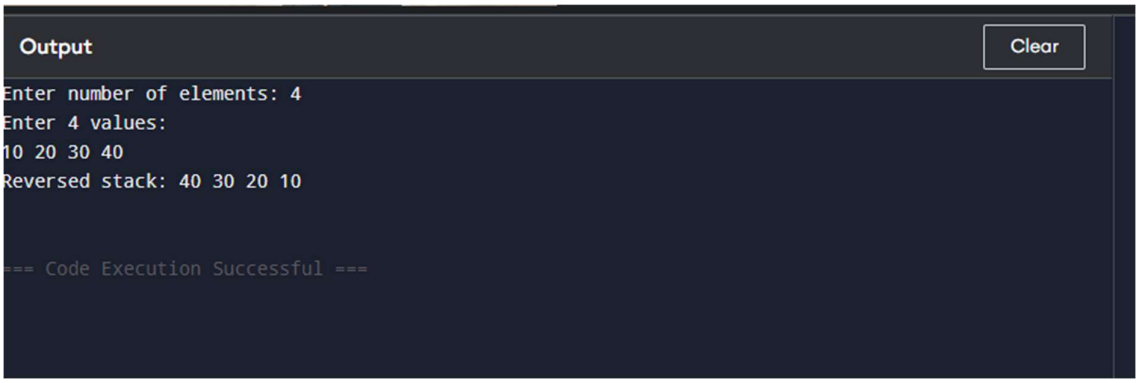
```
    }

    if (front == -1) front = 0; // first element

    rear++;
    queueArr[rear] = value;
}

// Stack push
void push(int value) {
    if (top == SIZE - 1) {
        cout << "Stack Overflow!\n";
        return;
    }
    top++;
    stackArr[top] = value;
}

// Stack pop
int pop() {
    if (top == -1) {
        cout << "Stack Underflow!\n";
        return -1;
    }
    int value = stackArr[top];
    top--;
    return value;
}
```

main.cpp

Share Run

```

int main() {
    int n, value;

    // Input
    cout << "Enter number of elements: ";
    cin >> n;

    cout << "Enter " << n << " values:\n";
    for (int i = 0; i < n; i++) {
        cin >> value;
        enqueue(value);
    }

    // Step 2: Move queue → stack (to reverse)
    for (int i = front; i <= rear; i++) {
        push(queueArr[i]);
    }

    // Step 3: Display reversed stack
    cout << "Reversed stack: ";
    while (top != -1) {
        cout << pop() << " ";
    }
    cout << endl;

    return 0;
}
```

CODE OUTPUT:



LAB TASK 4:

4. **Write a program to sort the elements of a stack.**

Algorithm:

SortStack Algorithm (Using Temporary Stack)

1. Create an empty temporary stack tempStack.
2. While original stack is NOT empty:
    - Pop the top element → call it temp
    - While tempStack is NOT empty AND tempStack.top() > temp → Pop from tempStack and push back to original stack
    - Push temp into tempStack
3. Copy all elements from tempStack back to original stack
4. Now the stack is sorted

CODE:

main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;

#define SIZE 20
int stackArr[SIZE];
int top = -1;

// Push
void push(int value) {
    if (top == SIZE - 1) {
        cout << "Stack Overflow!\n";
        return;
    }
    stackArr[++top] = value;
}

// Pop
int pop() {
    if (top == -1) {
        cout << "Stack Underflow!\n";
        return -1;
    }
    return stackArr[top--];
}

// Peek
int peek() {
    return stackArr[top];
```

main.cpp

Share Run

```
}

// Check if empty
bool isEmpty() {
    return top == -1;
}

// Sort stack using a temporary stack
void sortStack() {
    int tempStack[SIZE];
    int tempTop = -1;

    while (!isEmpty()) {
        int temp = pop();

        while (tempTop != -1 && tempStack[tempTop] > temp) {
            push(tempStack[tempTop--]);
        }

        tempStack[++tempTop] = temp;
    }

    // Move sorted values back to original stack
    while (tempTop != -1) {
        push(tempStack[tempTop--]);
    }
}
```

main.cpp

Share Run

```
void display() {
    if (top == -1) {
        cout << "Stack is empty.\n";
        return;
    }

    cout << "Stack: ";
    for (int i = top; i >= 0; i--) {
        cout << stackArr[i] << " ";
    }
    cout << endl;
}

int main() {
    int n, value;

    cout << "Enter number of elements: ";
    cin >> n;

    cout << "Enter " << n << " values:\n";
    for (int i = 0; i < n; i++) {
        cin >> value;
        push(value);
    }

    cout << "\nOriginal stack: ";
    display();
```

```
86      sortStack();
87
88      cout << "Sorted stack: ";
89      display();
90
91      return 0;
92  }
93
```

CODE OUTPUT:

```
Output                                                    Clear
Enter number of elements: 8
Enter 8 values:
40 10 50 20 5 60 30 15

Original stack: Stack: 15 30 60 5 20 50 10 40
Sorted stack: Stack: 5 10 15 20 30 40 50 60


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

LAB TASK 5: 5. Write a program to find the mean, variance and standard deviation of all elements of a stack

Algorithm

Algorithm: Mean / Variance / Standard Deviation

1. Start
2. Insert values into stack
3. Compute sum of all stack elements
4. Compute mean:

$$\text{mean} = \frac{\text{sum}}{\text{number of elements}}$$

5. For variance:
    - For each value x, compute

$$(x - \text{mean})^2$$

    - Sum all of these

- Divide by total count

6. Standard deviation =

$$\sqrt{\text{variance}}$$

7. Display mean, variance, standard deviation
8. End

CODE:

main.cpp

Share Run

```
    }
    cout << endl;
}

// Function to calculate mean
double findMean() {
    if (top == -1) return 0;

    double sum = 0;
    for (int i = 0; i <= top; i++)
        sum += stackArr[i];

    return sum / (top + 1);
}

// Function to calculate variance
double findVariance(double mean) {
    if (top == -1) return 0;

    double sumSquares = 0;

    for (int i = 0; i <= top; i++) {
        sumSquares += pow(stackArr[i] - mean, 2);
    }

    return sumSquares / (top + 1);
}
```

main.cpp

Share Run

```

// Function to calculate standard deviation
double findStdDev(double variance) {
    return sqrt(variance);
}

int main() {
    int n, value;

    cout << "Enter number of elements: ";
    cin >> n;

    cout << "Enter " << n << " values:\n";
    for (int i = 0; i < n; i++) {
        cin >> value;
        push(value);
    }

    display();

    double mean = findMean();
    double variance = findVariance(mean);
    double stdDev = findStdDev(variance);

    cout << "\nMean = " << mean << endl;
    cout << "Variance = " << variance << endl;
    cout << "Standard Deviation = " << stdDev << endl;

```

```
    cout << "\nMean = " << mean << endl;
    cout << "Variance = " << variance << endl;
    cout << "Standard Deviation = " << stdDev << endl;

    return 0;
}
```

CODE OUTPUT: