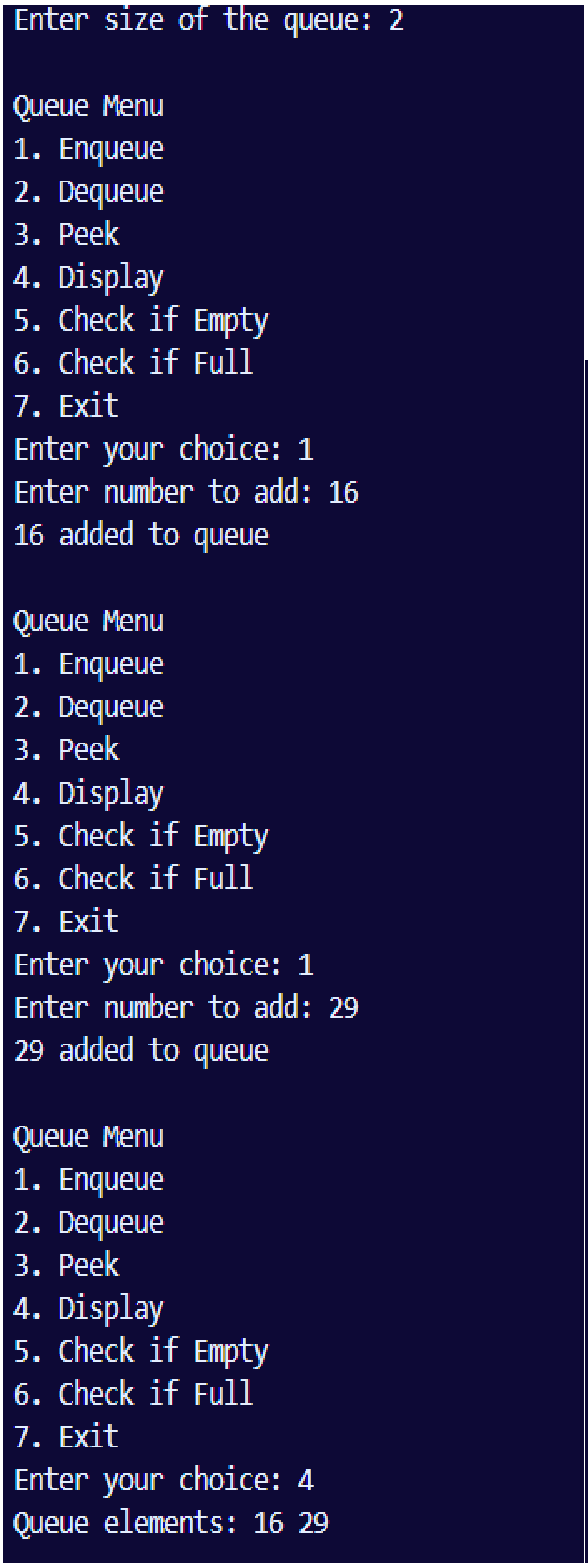
AQ1.cpp

```
#include <iostream>

using namespace std;


class Queue {


    int *data;

    int capacity;

    int frontIdx, backIdx;


public:

    Queue(int size) {

        capacity = size;

        data = new int[capacity];

        frontIdx = -1;

        backIdx = -1;

    }


    ~Queue() {

        delete[] data;

    }


    bool isEmpty() {

        return (frontIdx == -1 && backIdx == -1);

    }


    bool isFull() {
```

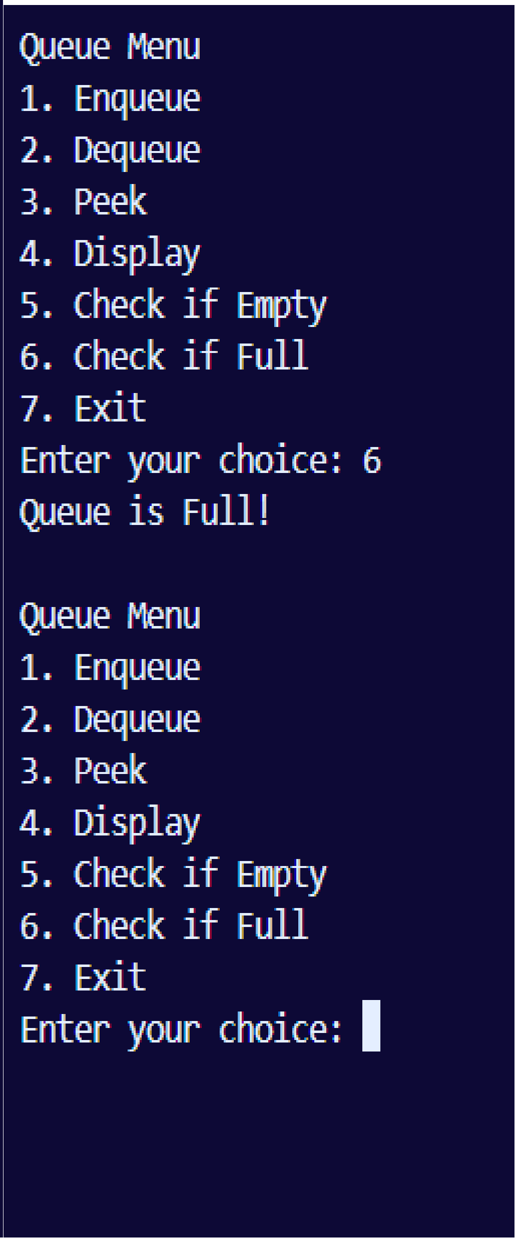
```
    return (backIdx == capacity - 1 && frontIdx == 0) || (backIdx + 1 == frontIdx);
}

void enqueue(int value) {
    if (isFull()) {
        cout << "Queue is Full!" << endl;
        return;
    }
    if (isEmpty()) {
        frontIdx = 0;
    }
    backIdx = (backIdx + 1) % capacity;
    data[backIdx] = value;
    cout << value << " added to queue" << endl;
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue is Empty!" << endl;
        return;
    }
    cout << data[frontIdx] << " removed from queue" << endl;
    if (frontIdx == backIdx) {
        frontIdx = backIdx = -1;
    } else {
        frontIdx = (frontIdx + 1) % capacity;
    }
}
```

```
    void peek() {
        if (isEmpty()) {
            cout << "Queue is Empty!" << endl;
        } else {
            cout << "Front element is: " << data[frontIdx] << endl;
        }
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is Empty!" << endl;
        } else {
            cout << "Queue elements: ";
            int i = frontIdx;
            while (i != backIdx) {
                cout << data[i] << " ";
                i = (i + 1) % capacity;
            }
            cout << data[backIdx] << endl;
        }
    }
};

int main() {
    int choice, value, size;
    cout << "Enter size of the queue: ";
    cin >> size;
```

```
Queue q(size);

do {
    cout << "\nQueue Menu\n";
    cout << "1. Enqueue\n";
    cout << "2. Dequeue\n";
    cout << "3. Peek\n";
    cout << "4. Display\n";
    cout << "5. Check if Empty\n";
    cout << "6. Check if Full\n";
    cout << "7. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
    case 1:
        cout << "Enter number to add: ";
        cin >> value;
        q.enqueue(value);
        break;
    case 2:
        q.dequeue();
        break;
    case 3:
        q.peek();
        break;
    case 4:
            q.display();
            break;
        case 5:
            if (q.isEmpty())
                cout << "Queue is Empty!" << endl;
            else
                cout << "Queue is NOT Empty!" << endl;
            break;
        case 6:
            if (q.isFull())
                cout << "Queue is Full!" << endl;
            else
                cout << "Queue is NOT Full!" << endl;
            break;
        case 7:
            cout << "Exiting program" << endl;
            break;
        default:
            cout << "Invalid choice, try again!" << endl;

    } while (choice != 7);

    return 0;
}
```



```
Enter size of the queue: 2

Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to add: 16
16 added to queue

Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to add: 29
29 added to queue

Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 4
Queue elements: 16 29
```



```
Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 6
Queue is Full!

Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice:
```

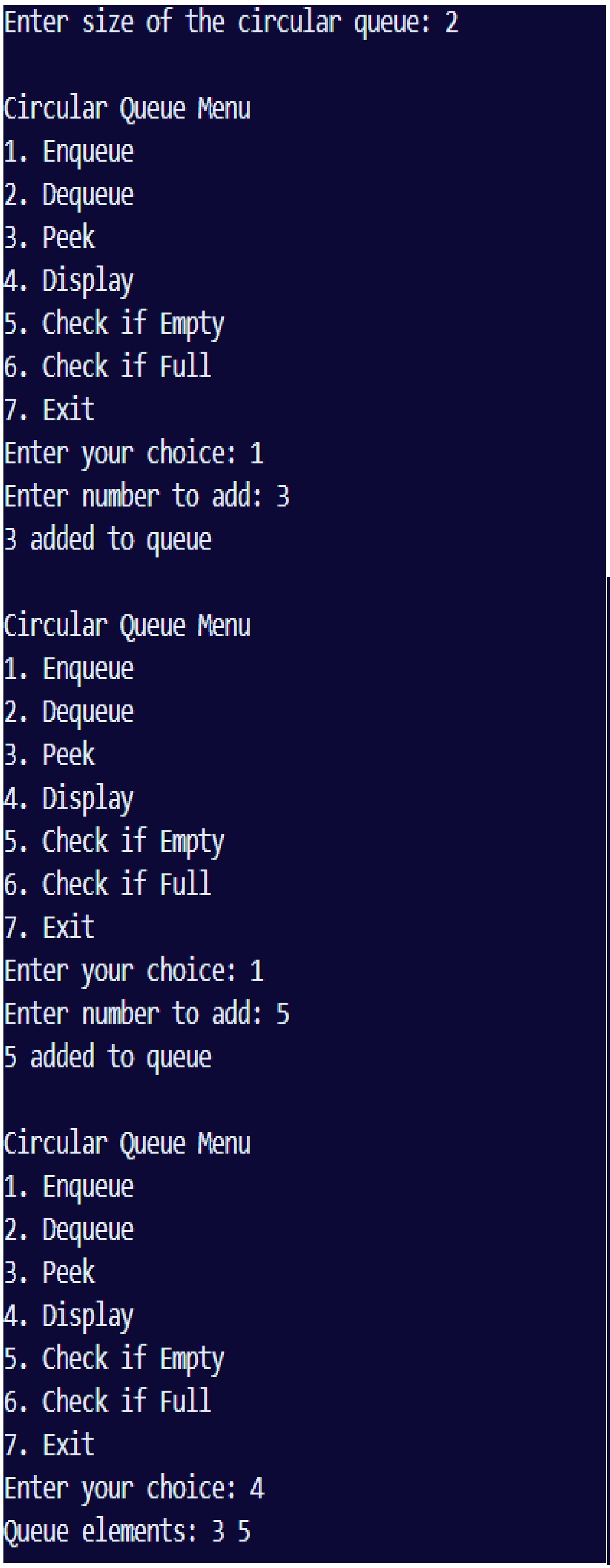
AQ2.cpp

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

class CircularQueue {
private:
    int *queueArray;
    int capacity;
    int frontIdx, backIdx;

public:
    CircularQueue(int size) {
        capacity = size;
        queueArray = new int[capacity];
        frontIdx = -1;
        backIdx = -1;
    }

    ~CircularQueue() {
        delete[] queueArray;
    }

    bool isEmpty() {
        return (frontIdx == -1 && backIdx == -1);
    }

    bool isFull() {
```

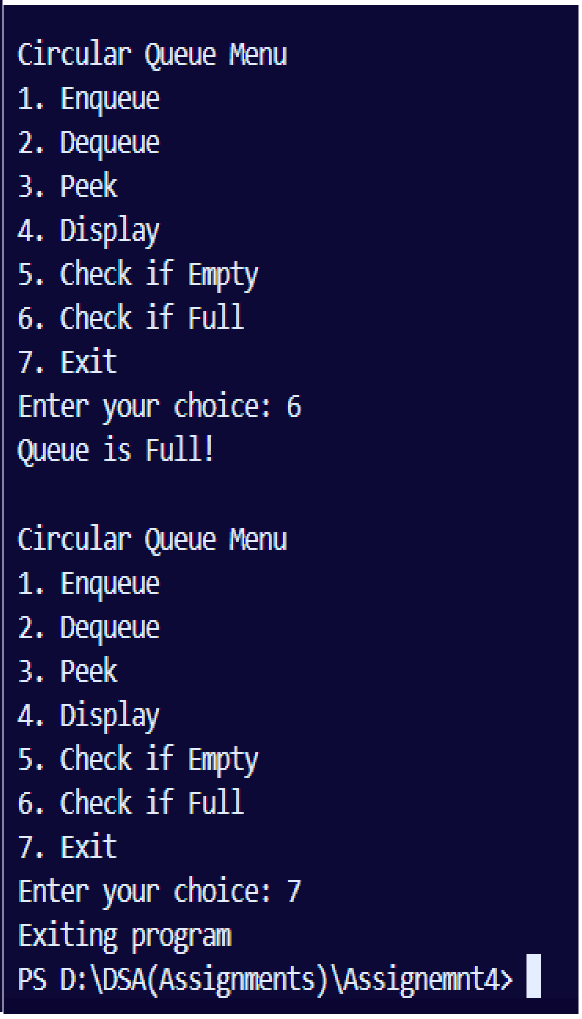
```
    return ((backIdx + 1) % capacity == frontIdx);
}

void enqueue(int value) {
    if (isFull()) {
        cout << "Queue is Full!" << endl;
        return;
    }
    if (isEmpty()) {
        frontIdx = backIdx = 0;
    } else {
        backIdx = (backIdx + 1) % capacity;
    }
    queueArray[backIdx] = value;
    cout << value << " added to queue" << endl;
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue is Empty!" << endl;
        return;
    }
    cout << queueArray[frontIdx] << " removed from queue" << endl;
    if (frontIdx == backIdx) {
        frontIdx = backIdx = -1;
    } else {
        frontIdx = (frontIdx + 1) % capacity;
    }
    }

    void peek() {
        if (isEmpty()) {
            cout << "Queue is Empty!" << endl;
        } else {
            cout << "Front element is: " << queueArray[frontIdx] << endl;
        }
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is Empty!" << endl;
        } else {
            cout << "Queue elements: ";
            int i = frontIdx;
            while (true) {
                cout << queueArray[i] << " ";
                if (i == backIdx) break;
                i = (i + 1) % capacity;
            }
            cout << endl;
        }
    }
};

int main() {
    int choice, value, size;
```

```
cout << "Enter size of the circular queue: ";
cin >> size;

CircularQueue q(size);

do {
    cout << "\nCircular Queue Menu\n";
    cout << "1. Enqueue\n";
    cout << "2. Dequeue\n";
    cout << "3. Peek\n";
    cout << "4. Display\n";
    cout << "5. Check if Empty\n";
    cout << "6. Check if Full\n";
    cout << "7. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
    case 1:
        cout << "Enter number to add: ";
        cin >> value;
        q.enqueue(value);
        break;
    case 2:
        q.dequeue();
        break;
    case 3:
        q.peek();
```

```
            break;
        case 4:
            q.display();
            break;
        case 5:
            if (q.isEmpty())
                cout << "Queue is Empty!" << endl;
            else
                cout << "Queue is NOT Empty!" << endl;
            break;
        case 6:
            if (q.isFull())
                cout << "Queue is Full!" << endl;
            else
                cout << "Queue is NOT Full!" << endl;
            break;
        case 7:
            cout << "Exiting program" << endl;
            break;
        default:
            cout << "Invalid choice, try again!" << endl;
        }
    } while (choice != 7);

    return 0;
}
```



```
Enter size of the circular queue: 2

Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to add: 3
3 added to queue

Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 1
Enter number to add: 5
5 added to queue

Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 4
Queue elements: 3 5
```

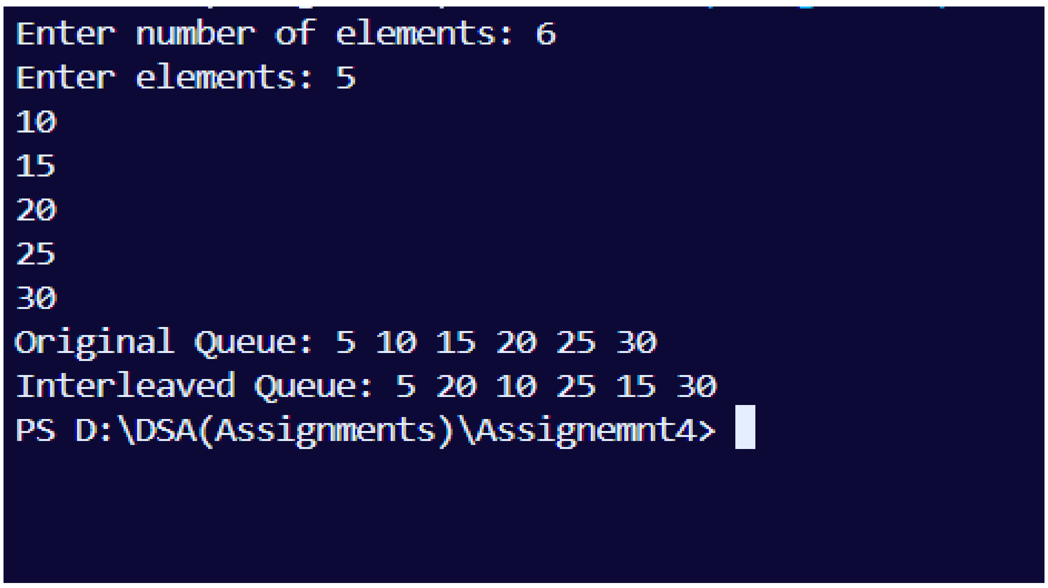


```
Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 6
Queue is Full!

Circular Queue Menu
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if Empty
6. Check if Full
7. Exit
Enter your choice: 7
Exiting program
PS D:\DSA(Assignments)\Assignemnt4>
```

AQ3.cpp

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

void interleaveQueue(queue<int> &q) {
   int n = q.size();
   if (n % 2 != 0) {
      cout << "Queue has an odd number of elements, cannot interleave!" << endl;
      return;
   }

   int half = n / 2;
   queue<int> firstHalf;

   for (int i = 0; i < half; i++) {
      firstHalf.push(q.front());
      q.pop();
   }

   \
   while (!firstHalf.empty()) {
      q.push(firstHalf.front());
      firstHalf.pop();
      q.push(q.front());
      q.pop();
   }
```

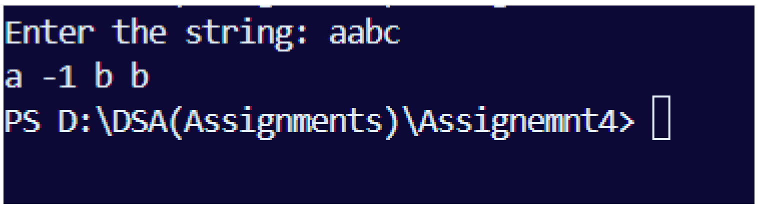
```
}

void displayQueue(queue<int> q) {
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    cout << endl;
}

int main() {
    queue<int> q;
    int n, num;

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

    cout << "Enter elements: ";
    for (int i = 0; i < n; i++) {
        cin >> num;
        q.push(num);
    }

    cout << "Original Queue: ";
    displayQueue(q);

    interleaveQueue(q);
    cout << "Interleaved Queue: ";
    displayQueue(q);

    return 0;
}
```



```
Enter number of elements: 6
Enter elements: 5
10
15
20
25
30
Original Queue: 5 10 15 20 25 30
Interleaved Queue: 5 20 10 25 15 30
PS D:\DSA(Assignments)\Assignemnt4>
```

AQ4.cpp

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

int main() {
    string s;
    cout << "Enter the string: ";
    cin >> s;
    queue<char> q;
    int count[26] = {0};

    for (int i = 0; i < s.length(); i++) {
        char c = s[i];
        count[c - 'a']++;
        q.push(c);

        while (!q.empty() && count[q.front() - 'a'] > 1) {
            q.pop();
        }

        if (q.empty()) {
            cout << "-1 ";
        } else {
            cout << q.front() << " ";
        }
    }

    return 0;
}
```

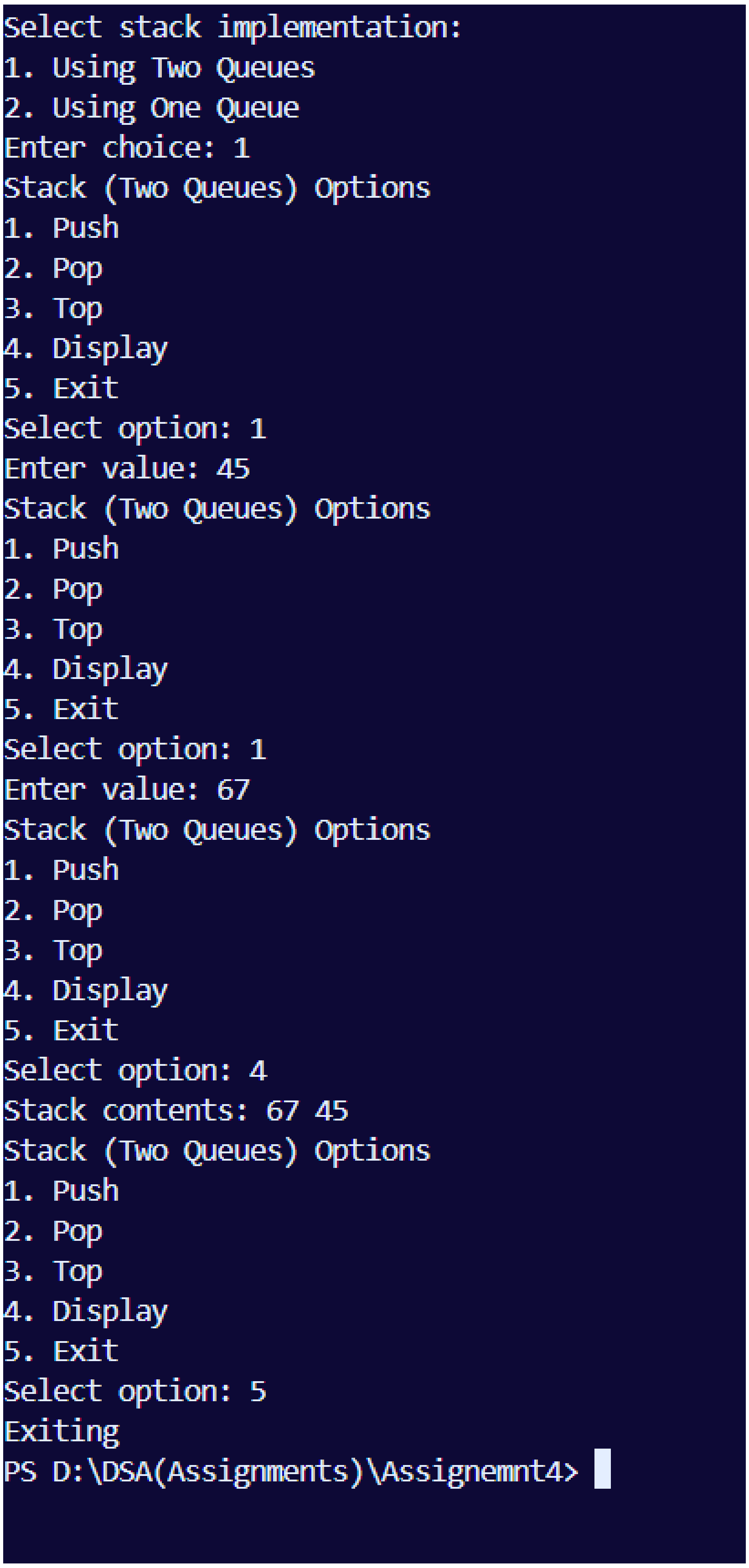


```
Enter the string: aabc
a -1 b b
PS D:\DSA(Assignments)\Assignemnt4>
```

AQ5.cpp

```
#include <iostream>

#include <queue>

using namespace std;


queue<int> q1, q2;

queue<int> q;


void stackPushTwoQueues(int element) {

   q2.push(element);

   while (!q1.empty()) {

      q2.push(q1.front());

      q1.pop();

   }

   swap(q1, q2);

}


void stackPopTwoQueues() {

   if(q1.empty()) {

      cout << "The stack is empty" << endl;

      return;

   }

   cout << q1.front() << " removed from stack" << endl;

   q1.pop();

}


int stackTopTwoQueues() {
    if(q1.empty()) {
        cout << "The stack is empty" << endl;
        return -1;
    }
    return q1.front();
}

void stackDisplayTwoQueues() {
    if(q1.empty()) {
        cout << "The stack is empty" << endl;
        return;
    }
    queue<int> tempQueue = q1;
    cout << "Stack contents: ";
    while (!tempQueue.empty()) {
        cout << tempQueue.front() << " ";
        tempQueue.pop();
    }
    cout << endl;
}

void stackPushOneQueue(int element) {
    int size = q.size();
    q.push(element);
    for (int i = 0; i < size; i++) {
        q.push(q.front());
        q.pop();
    }
```

```
}

void stackPopOneQueue() {
    if(q.empty()) {
        cout << "The stack is empty" << endl;
        return;
    }
    cout << q.front() << " removed from stack" << endl;
    q.pop();
}

int stackTopOneQueue() {
    if(q.empty()) {
        cout << "The stack is empty" << endl;
        return -1;
    }
    return q.front();
}

void stackDisplayOneQueue() {
    if(q.empty()) {
        cout << "The stack is empty" << endl;
        return;
    }
    queue<int> tempQueue = q;
    cout << "Stack contents: ";
    while (!tempQueue.empty()) {
        cout << tempQueue.front() << " ";
        tempQueue.pop();
    }
    cout << endl;
}

int main() {
    int stackType, choice, value;
    cout << "Select stack implementation:\n1. Using Two Queues\n2. Using One
Queue\nEnter choice: ";
    cin >> stackType;

    if(stackType == 1) {
        do {
            cout << "Stack (Two Queues) Options\n";
            cout << "1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\nSelect option: ";
            cin >> choice;
            switch (choice) {
                case 1:
                    cout << "Enter value: ";
                    cin >> value;
                    stackPushTwoQueues(value);
                    break;
                case 2:
                    stackPopTwoQueues();
                    break;
                case 3:
                    cout << "Top value: " << stackTopTwoQueues() << endl;
                    break;
                case 4:
                    stackDisplayTwoQueues();
                    break;
                case 5:
                    cout << "Exiting" << endl;
                    break;
                default:
                    cout << "Invalid selection" << endl;
            }
        } while (choice != 5);
    } else if(stackType == 2) {
        do {
            cout << "Stack (One Queue) Options\n";
            cout << "1. Push\n2. Pop\n3. Top\n4. Display\n5. Exit\nSelect option: ";
            cin >> choice;
            switch (choice) {
                case 1:
                    cout << "Enter value: ";
                    cin >> value;
                    stackPushOneQueue(value);
                    break;
                case 2:
                    stackPopOneQueue();
                    break;
                case 3:
                    cout << "Top value: " << stackTopOneQueue() << endl;
                    break;
                case 4:
                    stackDisplayOneQueue();
                    break;
                case 5:
                    cout << "Exiting" << endl;
                    break;
                default:
                    cout << "Invalid selection" << endl;
            }
        } while (choice != 5);
    } else {
        cout << "Invalid stack implementation choice" << endl;
    }
    return 0;
    }
```



```
Select stack implementation:
1. Using Two Queues
2. Using One Queue
Enter choice: 1
Stack (Two Queues) Options
1. Push
2. Pop
3. Top
4. Display
5. Exit
Select option: 1
Enter value: 45
Stack (Two Queues) Options
1. Push
2. Pop
3. Top
4. Display
5. Exit
Select option: 1
Enter value: 67
Stack (Two Queues) Options
1. Push
2. Pop
3. Top
4. Display
5. Exit
Select option: 4
Stack contents: 67 45
Stack (Two Queues) Options
1. Push
2. Pop
3. Top
4. Display
5. Exit
Select option: 5
Exiting
PS D:\DSA(Assignments)\Assignemnt4>
```