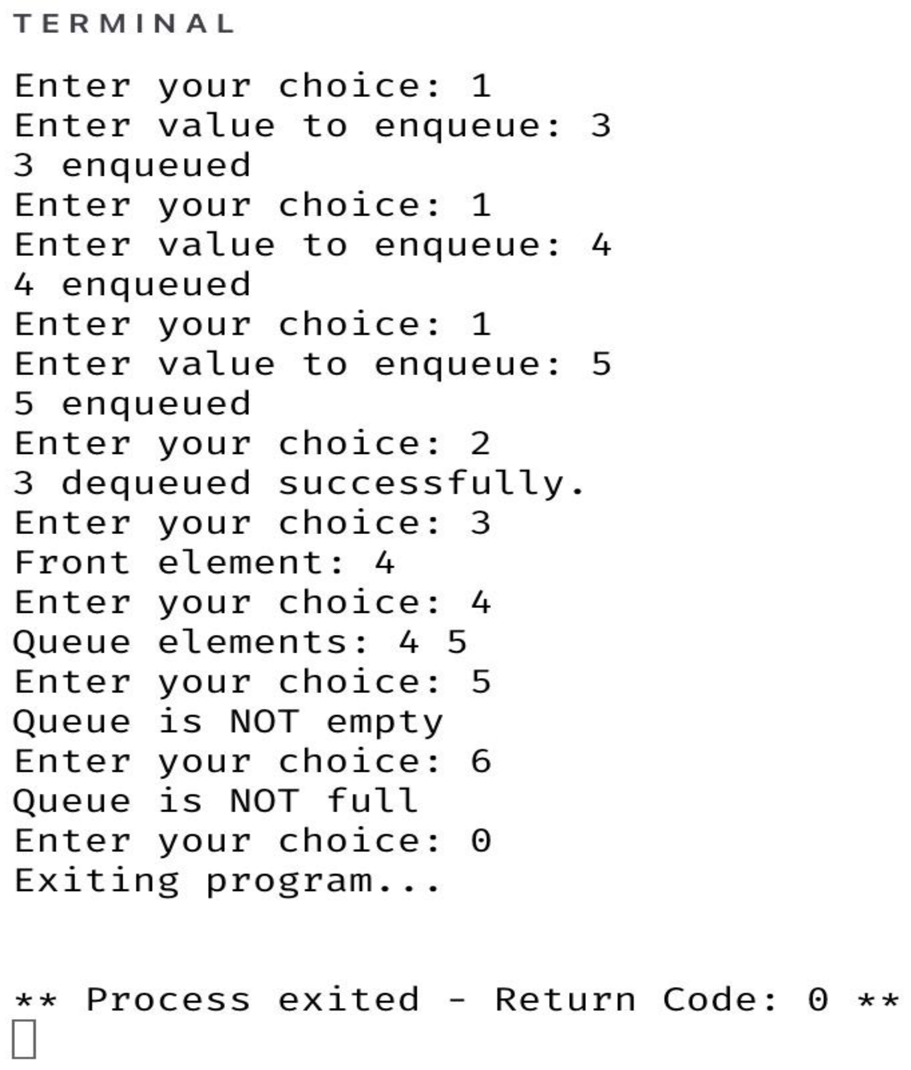
DATA STRUCTURES AND ALGORITHMS –> ASSIGNMENT - 4
AADITA GARG -> 1024030461 -> 2C32

(1) Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

```
#include <iostream>
using namespace std;
#define MAX 5

class Queue {
   int arr[MAX];
   int rear;
   int front;
public:
   Queue() {
      rear = -1;
      front = -1;
   }

   bool isEmpty() {
      return (front == -1);
   }

   bool isFull() {
      return (rear == MAX - 1);
   }

   void enqueue(int x) {
      if (isFull()) {
         cout << "Queue is full! Cannot enqueue " << x << endl;
         return;
      }
      if (front == -1) front = 0; // first element
      arr[++rear] = x;
      cout << x << " enqueued" << endl;
   }

   void dequeue() {
      if (isEmpty()) {
         cout << "Queue is empty!" << endl;
         return;
      }
      cout << arr[front] << " dequeued successfully." << endl;

      if (front == rear) {
         // Queue becomes empty after removal
```

```
            front = rear = -1;
        } else {
            front++;
        }
    }

    void peek() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return;
        }
        cout << "Front element: " << arr[front] << endl;
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is empty!" << endl;
            return;
        }
        cout << "Queue elements: ";
        for (int i = front; i <= rear; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    Queue q;
    int choice, value;

    do {
        cout << "Enter your choice: ";
        cin >> choice;

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

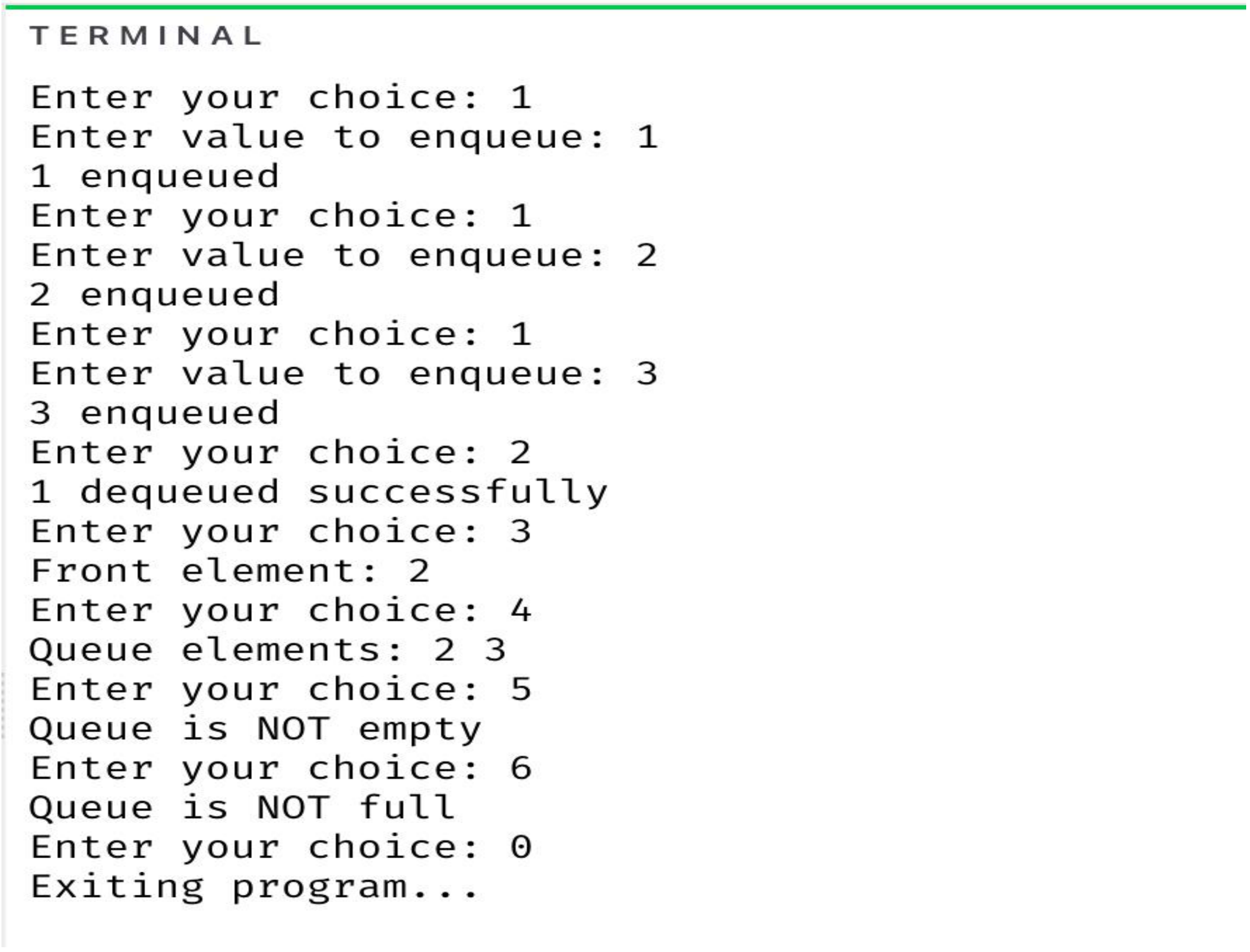
```
          case 4:
            q.display();
        break;
         case 5:
            cout << (q.isEmpty() ? "Queue is EMPTY" : "Queue is NOT empty") << endl;
            break;
         case 6:
            cout << (q.isFull() ? "Queue is FULL" : "Queue is NOT full") << endl;
            break;
         case 0:
            cout << "Exiting program..." << endl;
            break;
         default:
            cout << "Invalid choice! Try again." << endl;
      }
   } while (choice != 0);

   return 0;
}
```


```
TERMINAL

Enter your choice: 1
Enter value to enqueue: 3
3 enqueued
Enter your choice: 1
Enter value to enqueue: 4
4 enqueued
Enter your choice: 1
Enter value to enqueue: 5
5 enqueued
Enter your choice: 2
3 dequeued successfully.
Enter your choice: 3
Front element: 4
Enter your choice: 4
Queue elements: 4 5
Enter your choice: 5
Queue is NOT empty
Enter your choice: 6
Queue is NOT full
Enter your choice: 0
Exiting program...


** Process exited - Return Code: 0 **
```

(2) Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

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

#define MAX 5

class CircularQueue {
   int arr[MAX];
   int front;
   int rear;

public:
   CircularQueue() {
      front = -1;
      rear = -1;
   }

   bool isEmpty() {
      return (front == -1);
   }

   bool isFull() {
      return ((rear + 1) % MAX == front);
   }

   void enqueue(int x) {
      if (isFull()) {
         cout << "Queue is FULL! Cannot enqueue " << x << endl;
         return;
      }
      if (isEmpty()) {
         front = rear = 0;
      } else {
         rear = (rear + 1) % MAX;
      }
      arr[rear] = x;
      cout << x << " enqueued" << endl;
   }

   void dequeue() {
      if (isEmpty()) {
         cout << "Queue is EMPTY! Cannot dequeue" << endl;
         return;
      }
      cout << arr[front] << " dequeued successfully" << endl;
      if (front == rear) {
         // Queue has only one element, becomes empty
```

```
            front = rear = -1;
        } else {
            front = (front + 1) % MAX;
        }
    }

    void peek() {
        if (isEmpty()) {
            cout << "Queue is EMPTY!" << endl;
            return;
        }
        cout << "Front element: " << arr[front] << endl;
    }

    void display() {
        if (isEmpty()) {
            cout << "Queue is EMPTY!" << endl;
            return;
        }
        cout << "Queue elements: ";
        int i = front;
        while (true) {
            cout << arr[i] << " ";
            if (i == rear) break;
            i = (i + 1) % MAX;
        }
        cout << endl;
    }
};

int main() {
    CircularQueue q;
    int choice, value;

    do {
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to enqueue: ";
                cin >> value;
                q.enqueue(value);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.peek();
                break;
            case 4:
                q.display();
                break;
            case 5:
                cout << (q.isEmpty() ? "Queue is EMPTY" : "Queue is NOT empty") << endl;
                break;
            case 6:
                cout << (q.isFull() ? "Queue is FULL" : "Queue is NOT full") << endl;
                break;
            case 0:
                cout << "Exiting program..." << endl;
                break;
            default:
                cout << "Invalid choice! Try again." << endl;
        }
    } while (choice != 0);

    return 0;
}
```



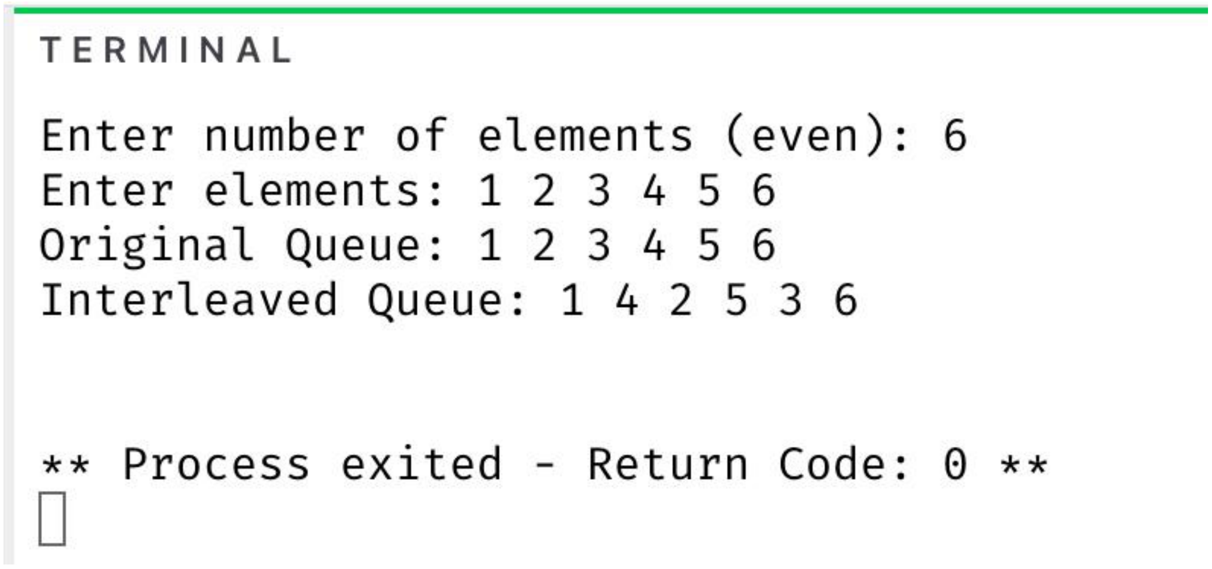
TERMINAL

```
Enter your choice: 1
Enter value to enqueue: 1
1 enqueued
Enter your choice: 1
Enter value to enqueue: 2
2 enqueued
Enter your choice: 1
Enter value to enqueue: 3
3 enqueued
Enter your choice: 2
1 dequeued successfully
Enter your choice: 3
Front element: 2
Enter your choice: 4
Queue elements: 2 3
Enter your choice: 5
Queue is NOT empty
Enter your choice: 6
Queue is NOT full
Enter your choice: 0
Exiting program...
```

(3) Write a program interleave the first half of the queue with second half.
Sample I/P: 4 7 11 20 5 9 Sample O/P: 4 20 7 5 11 9

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

void interleaveQueue(queue<int>& q) {
  int n = q.size();

  // If queue has odd number of elements, cannot interleave evenly
  if (n % 2 != 0) {
    cout << "Queue has odd number of elements, cannot interleave!" << endl;
    return;
  }

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

  // Move first half elements into another queue
  for (int i = 0; i < half; i++) {
    firstHalf.push(q.front());
    q.pop();
  }

  // Now interleave both halves
  while (!firstHalf.empty()) {
    // Take from first half
    q.push(firstHalf.front());
    firstHalf.pop();

    // Take from second half
    q.push(q.front());
    q.pop();
  }
}

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

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

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

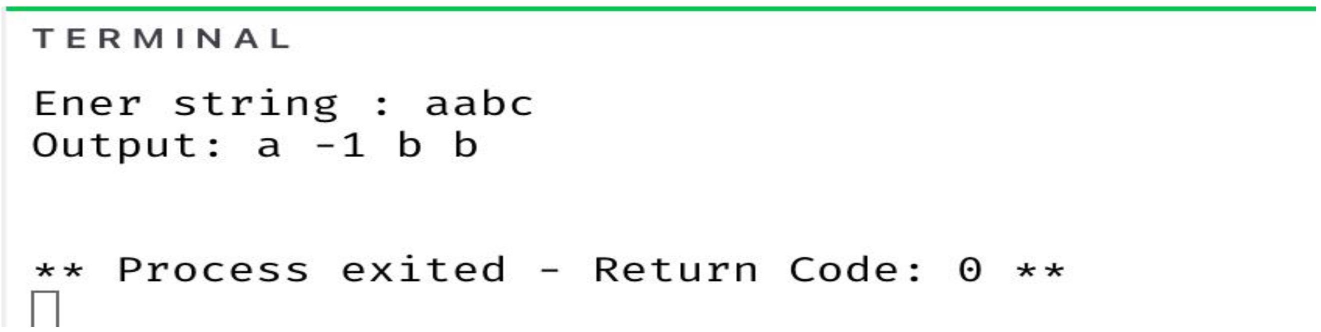
```
    cout << "Original Queue: ";
    {
        queue<int> temp = q;
        while (!temp.empty()) {
            cout << temp.front() << " ";
            temp.pop();
        }
        cout << endl;
    }

    interleaveQueue(q);

    cout << "Interleaved Queue: ";
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    cout << endl;

    return 0;
}
```



```
Enter number of elements (even): 6
Enter elements: 1 2 3 4 5 6
Original Queue: 1 2 3 4 5 6
Interleaved Queue: 1 4 2 5 3 6


** Process exited - Return Code: 0 **
```

(4) Write a program to find first non-repeating character in a string using Queue.
Sample I/P: a a b c Sample O/P: a -1 b b

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

void firstNonRepeating(string str) {
   queue<char> q;
   int freq[256] = {0};

   for (char ch : str) {
      if (ch == ' ') continue;

      freq[ch]++;
      q.push(ch);

      while (!q.empty() && freq[q.front()] > 1) {
         q.pop();
      }

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

int main() {
   string input;
   cout << "Ener string : ";
   getline(cin, input);

   cout << "Output: ";
   firstNonRepeating(input);

   return 0;
}
```

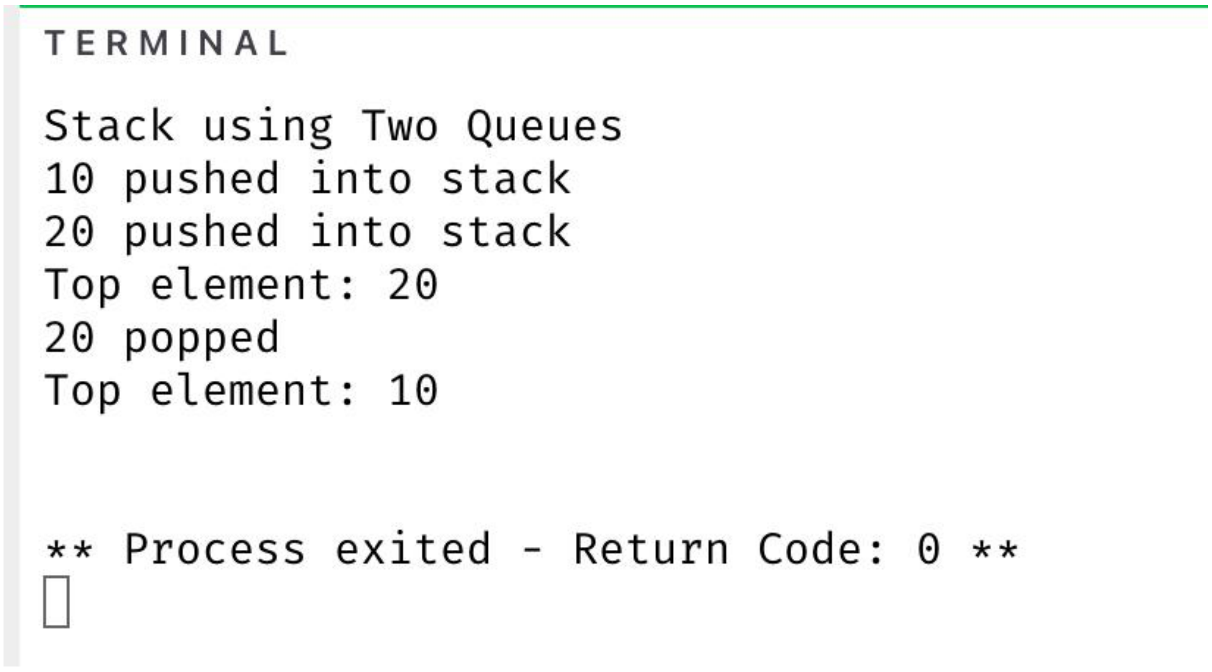


```
TERMINAL

Ener string : aabc
Output: a -1 b b


** Process exited - Return Code: 0 **
```

(5) (A) Write a program to implement a stack using two queues

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

class StackTwoQueues {
   queue<int> q1, q2;

public:
   void push(int x) {
      q1.push(x);
      cout << x << " pushed into stack\n";
   }

   void pop() {
      if (q1.empty()) {
         cout << "Stack is EMPTY\n";
         return;
      }

      while (q1.size() > 1) {
         q2.push(q1.front());
         q1.pop();
      }

      cout << q1.front() << " popped\n";
      q1.pop();

      // Swap q1 and q2
      swap(q1, q2);
   }

   void top() {
      if (q1.empty()) {
         cout << "Stack is EMPTY\n";
         return;
      }
      cout << "Top element: " << q1.back() << endl;
   }

   bool empty() {
      return q1.empty();
   }
};
```

```
    int main() {
    StackTwoQueues s1;

    cout << "Stack using Two Queues\n";
    s1.push(10);
    s1.push(20);
    s1.top();
    s1.pop();
    s1.top();
    return 0;
}
```



```
Stack using Two Queues
10 pushed into stack
20 pushed into stack
Top element: 20
20 popped
Top element: 10


** Process exited - Return Code: 0 **
```

(5) (B) Write a program to implement a stack using one queue

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

class StackOneQueue {
   queue<int> q;
public:
   void push(int x) {
      q.push(x);
      int size = q.size();
      for (int i = 0; i < size - 1; i++) {
         q.push(q.front());
         q.pop();
      }
      cout << x << " pushed into stack\n";
   }
   void pop() {
      if (q.empty()) {
         cout << "Stack is empty\n";
         return;
      }
      cout << q.front() << " popped\n";
      q.pop();
   }
   void top() {
      if (q.empty()) {
         cout << "Stack is empty\n";
         return;
      }
      cout << "Top element: " << q.front() << endl;
   }
   bool empty() { return q.empty(); }
};

int main() {
   StackOneQueue s2;

   cout << "Stack using One Queue\n";
   s2.push(5);
   s2.push(15);
   s2.top();
   s2.pop();
   s2.top();

   return 0;
}
```

TERMINAL

```
Stack using One Queue
5 pushed into stack
15 pushed into stack
Top element: 15
15 popped
Top element: 5
```