ASSIGNMENT 4 DATA STRUCTURES

Q1. Develop a menu driven program demonstrating the following operations on simple Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

Code :

Q1.cpp > ...

```cpp
//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 SIZE 5

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

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

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

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue is Full\n";
            return;
        }
        if (front == -1) front = 0;
        arr[++rear] = x;
        cout << x << " inserted\n";
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Queue is Empty\n";
```

Q1.cpp > ...

```cpp
class Queue {
    void dequeue() {
        if (isEmpty()) {
            return;
        }
        cout << arr[front++] << " removed\n";
    }

    void peek() {
        if (isEmpty()) cout << "Queue is Empty\n";
        else cout << "Front element: " << arr[front] << endl;
    }

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

int main() {
    Queue q;
    int ch, val;
    do {
        cout << "\n1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit\n";
        cin >> ch;
        switch (ch) {
        case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
        case 2: q.dequeue(); break;
        case 3: q.peek(); break;
        case 4: q.display(); break;
        }
    } while (ch != 5);
    return 0;
}
```

Output :

```
1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
1
1
Enter value: 23
23 inserted
Enter value: 23
23 inserted
23 inserted

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
1
Enter value: 55
55 inserted

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
4
Queue: 23 55

Enter value: 55
55 inserted

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
4
Queue: 23 55


1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
4
Queue: 23 55


1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
3
Front element: 23

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
2
23 removed

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
```

Q2. Develop a menu driven program demonstrating the following operations on Circular Queues: enqueue(), dequeue(), isEmpty(), isFull(), display(), and peek().

Code :

Q2.cpp > ...

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

#define SIZE 5

class CircularQueue {
    int arr[SIZE];
    int front, rear;
public:
    CircularQueue() {
        front = -1;
        rear = -1;
    }

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

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

    void enqueue(int x) {
        if (isFull()) {
            cout << "Queue is Full\n";
            return;
        }
        if (isEmpty())
            front = rear = 0;
        else
            rear = (rear + 1) % SIZE;
        arr[rear] = x;
        cout << x << " inserted\n";
    }

    void dequeue() {
        if (isEmpty()) {
```

Q2.cpp > ...

```
6   class CircularQueue {
36      void dequeue() {
37          if (isEmpty()) {
38              cout << "Queue is Empty\n";
39              return;
40          }
41          cout << arr[front] << " removed\n";
42          if (front == rear)
43              front = rear = -1;
44          else
45              front = (front + 1) % SIZE;
46      }
47
48      void peek() {
49          if (isEmpty()) cout << "Queue is Empty\n";
50          else cout << "Front element: " << arr[front] << endl;
51      }
52
53      void display() {
54          if (isEmpty()) {
55              cout << "Queue is Empty\n";
56              return;
57          }
58          cout << "Queue: ";
59          int i = front;
60          while (true) {
61              cout << arr[i] << " ";
62              if (i == rear) break;
63              i = (i + 1) % SIZE;
64          }
65          cout << endl;
66      }
67  };
68
69  int main() {
70      CircularQueue q;
71      int ch, val;
```

```
    int ch, val;
    do {
        cout << "\n1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit\n";
        cin >> ch;
        switch (ch) {
        case 1: cout << "Enter value: "; cin >> val; q.enqueue(val); break;
        case 2: q.dequeue(); break;
        case 3: q.peek(); break;
        case 4: q.display(); break;
        }
    } while (ch != 5);
    return 0;
}
```

Output :

```
1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
1
Enter value: 23
23 inserted

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
1
Enter value: 44
44 inserted

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
2
23 removed

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
3
Front element: 44

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
4
Queue: 44

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
3
Front element: 44

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
2
44 removed

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
3
Queue is Empty

1.Enqueue 2.Dequeue 3.Peek 4.Display 5.Exit
4
Queue is Empty
```

Q3. 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

Code :

Q3.cpp > ...

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

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

    for (int i = 0; i < n / 2; i++) {
        firstHalf.push(q.front());
        q.pop();
    }

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

int main() {
    queue<int> q;
    int n, val;
    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++) {
        cin >> val;
        q.push(val);
    }

    interleaveQueue(q);

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

```
38          q.pop();
39      }
40      cout << endl;
41      return 0;
42  }
```

Output :

```
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> g++ Q3.cpp -o Q3
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> g++ Q3.cpp -o Q3
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> ./Q3
Enter number of elements: 6
Enter elements: 4 7 11 20 5 9
Interleaved Queue: 4 20 7 5 11 9
```

Q4. 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

Code with output :

Q4.cpp > ...

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

int main() {
    string s;
    cout << "Enter characters (use space separated like a a b c): ";
    getline(cin, s);  // ✅ takes full line input

    queue<char> q;
    int freq[26] = {0};

    cout << "Output: ";

    for (int i = 0; i < s.length(); i++) {
        if (s[i] == ' ') continue; // skip spaces

        char c = s[i];
        freq[c - 'a']++;
        q.push(c);

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

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

    cout << endl;
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> ./Q4
Enter characters (use space separated like a a b c):  a a b c e e
Output: a -1 b b b b
```

Q5. Write a program to implement a stack using (a) Two queues and (b) One Queue.

Code :

a)

Q5A.cpp > ...

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

class Stack {
    queue<int> q1, q2;
public:
    void push(int x) {
        q2.push(x);
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
        swap(q1, q2);
    }

    void pop() {
        if (q1.empty()) cout << "Stack Empty\n";
        else q1.pop();
    }

    int top() {
        if (q1.empty()) return -1;
        return q1.front();
    }

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

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);
    cout << s.top() << endl; // 30
```

```
    s.pop();
    cout << s.top() << endl; // 20
    return 0;
}
```

Output :

```
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> ./Q5A
30
20
```

b) Using One queue

Code :

Q5B.cpp > ...

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

class Stack {
    queue<int> q;
public:
    void push(int x) {
        int n = q.size();
        q.push(x);
        for (int i = 0; i < n; i++) {
            q.push(q.front());
            q.pop();
        }
    }

    void pop() {
        if (q.empty()) cout << "Stack Empty\n";
        else q.pop();
    }

    int top() {
        if (q.empty()) return -1;
        return q.front();
    }
};

int main() {
    Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    cout << s.top() << endl; // 3
    s.pop();
    cout << s.top() << endl; // 2
    return 0;
}
```

Output :

```
PS C:\Users\hxrle\OneDrive\Dokumen\ASSIGNMENT 4 DS> ./Q5B
3
3
2
```