ASSIGNMENT 4

**QUESTION 1**

#include <iostream>

using namespace std;

class Queue {

    int front, rear, size;

    int\* arr;

public:

    Queue(int s) {

        size = s;

        arr = new int[size];

        front = -1;

        rear = -1;

    }

    bool isEmpty() {

        if (front == -1)

            return true;

        else

            return false;

    }

    bool isFull() {

        if (rear == size - 1)

            return true;

        else

            return false;

    }

    void enqueue(int x) {

        if (isFull()) {

            cout << "Queue is full"<<endl;

            return;

        }

        if (front == -1)

            front = 0;

        rear++;

        arr[rear] = x;

        cout << x << " enqueued"<<endl;

    }

    void dequeue() {

        if (isEmpty()) {

            cout << "Queue is empty"<<endl;

            return;

        }

        cout << arr[front] << " dequeued"<<endl;

        if (front == rear) {

            front = -1;

            rear = -1;

        } else {

            front++;

        }

    }

    void peek() {

        if (isEmpty()) {

            cout << "Queue is empty"<<endl;

        } else {

            cout << "Front element: " << arr[front] << " ";

        }

    }

    void display() {

        if (isEmpty()) {

            cout << "Queue is empty\n";

            return;

        }

        cout << "Queue elements: ";

        for (int i = front; i <= rear; i++) {

            cout << arr[i] << " ";

        }

        cout << "\n";

    }

};

int main() {

    int s;

    cout << "Enter size of queue: ";

    cin >> s;

    Queue q(s);

    int choice, val;

    bool running = true;

    while (running) {

        cout << "1.Enqueue\n2.Dequeue\n3.IsEmpty\n4.IsFull\n5.Display\n6.Peek\nAny other number to Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

            cout << "Enter value: ";

            cin >> val;

            q.enqueue(val);

            break;

        case 2:

            q.dequeue();

            break;

        case 3:

            if (q.isEmpty())

                cout << "Queue is empty\n";

            else

                cout << "Queue is not empty\n";

            break;

        case 4:

            if (q.isFull())

                cout << "Queue is full\n";

            else

                cout << "Queue is not full\n";

            break;

        case 5:

            q.display();

            break;

        case 6:

            q.peek();

            break;

        default:

            cout << "invalid"<<endl;

        }

    }

    return 0;

}

**QUESTION 2**

#include <iostream>

using namespace std;

class CircularQueue {

    int front, rear, size;

    int\* arr;

public:

    CircularQueue(int s) {

        size = s;

        arr = new int[size];

        front = -1;

        rear = -1;

    }

    bool isEmpty() {

        if (front == -1)

            return true;

        else

            return false;

    }

    bool isFull() {

        if ((front == 0 && rear == size - 1) || (rear + 1 == front))

            return true;

        else

            return false;

    }

    void enqueue(int x) {

        if (isFull()) {

            cout << "Queue is full\n";

            return;

        }

        if (front == -1) {

            front = 0;

            rear = 0;

        } else if (rear == size - 1 && front != 0) {

            rear = 0;

        } else {

            rear++;

        }

        arr[rear] = x;

        cout << x << " enqueued\n";

    }

    void dequeue() {

        if (isEmpty()) {

            cout << "Queue is empty\n";

            return;

        }

        cout << arr[front] << " dequeued\n";

        if (front == rear) {

            front = -1;

            rear = -1;

        } else if (front == size - 1) {

            front = 0;

        } else {

            front++;

        }

    }

    void peek() {

        if (isEmpty()) {

            cout << "Queue is empty\n";

        } else {

            cout << "Front element: " << arr[front] << "\n";

        }

    }

    void display() {

        if (isEmpty()) {

            cout << "Queue is empty\n";

            return;

        }

        cout << "Queue elements: ";

        if (rear >= front) {

            for (int i = front; i <= rear; i++)

                cout << arr[i] << " ";

        } else {

            for (int i = front; i < size; i++)

                cout << arr[i] << " ";

            for (int i = 0; i <= rear; i++)

                cout << arr[i] << " ";

        }

        cout << "\n";

    }

};

int main() {

    int s;

    cout << "Enter size of queue: ";

    cin >> s;

    CircularQueue q(s);

    int choice, val;

    bool running = true;

    while (running) {

        cout << "\n1.Enqueue\n2.Dequeue\n3.IsEmpty\n4.IsFull\n5.Display\n6.Peek\nAny other number to Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

            cout << "Enter value: ";

            cin >> val;

            q.enqueue(val);

            break;

        case 2:

            q.dequeue();

            break;

        case 3:

            if (q.isEmpty())

                cout << "Queue is empty\n";

            else

                cout << "Queue is not empty\n";

            break;

        case 4:

            if (q.isFull())

                cout << "Queue is full\n";

            else

                cout << "Queue is not full\n";

            break;

        case 5:

            q.display();

            break;

        case 6:

            q.peek();

            break;

        default:

            cout << "Exiting...\n";

            running = false;

        }

    }

    return 0;

}

**QUESTION 3**

#include <iostream>

#include <queue>

using namespace std;

void interleaveQueue(queue<int>& q) {

    int n = q.size();

    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();

    }

}

int main() {

    queue<int> q;

    int n, val;

    cout << "Enter even number of elements: ";

    cin >> n;

    if (n % 2 != 0) {

        cout << "Only even number of elements allowed\n";

        return 0;

    }

    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() << " ";

        q.pop();

    }

    cout << "\n";

    return 0;

}

**QUESTION 4**

#include <bits/stdc++.h>

using namespace std;

int main() {

    string s;

    cout << "Enter string (lowercase letters): ";

    cin >> s;

    queue<char> q;

    int freq[26] = {0};

    cout << "Output: ";

    for (char c : s) {

        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;

}

**QUESTION 5(a)**

#include <bits/stdc++.h>

using namespace std;

class StackTwoQueues {

    queue<int> q1, q2;

public:

    void push(int x) {

        q2.push(x);

        while (!q1.empty()) {

            q2.push(q1.front());

            q1.pop();

        }

        swap(q1, q2);

        cout << x << " pushed to stack" << endl;

    }

    void pop() {

        if (q1.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

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

        q1.pop();

    }

    void top() {

        if (q1.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << "Top element: " << q1.front() << endl;

    }

    void display() {

        if (q1.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << "Stack elements (top to bottom): ";

        queue<int> temp = q1;

        while (!temp.empty()) {

        cout << temp.front() << " ";

            temp.pop();

        }

        cout << endl;

    }

    bool empty() {

        if (q1.empty())

            return true;

        else

            return false;

    }

};

int main() {

    StackTwoQueues st;

    int choice, val;

    bool running = true;

    while (running) {

        cout << "\n1.Push\n2.Pop\n3.Top\n4.Display\n5.Check Empty\nAny other number to Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter value: ";

            cin >> val;

            st.push(val);

        } else if (choice == 2) {

            st.pop();

        } else if (choice == 3) {

            st.top();

        } else if (choice == 4) {

            st.display();

        } else if (choice == 5) {

            if (st.empty())

                cout << "Stack is empty" << endl;

            else

                cout << "Stack is not empty" << endl;

        } else {

            cout << "Exit." << endl;

            running = false;

        }

    }

    return 0;

}

**5(B)**#include <bits/stdc++.h>

using namespace std;

class StackOneQueue {

    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();

        }

        cout << x << " pushed to stack" << endl;

    }

    void pop() {

        if (q.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << q.front() << " popped from stack" << endl;

        q.pop();

    }

    void top() {

        if (q.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << "Top element: " << q.front() << endl;

    }

    void display() {

        if (q.empty()) {

            cout << "Stack is empty" << endl;

            return;

        }

        cout << "Stack elements (top to bottom): ";

        queue<int> temp = q;

        while (!temp.empty()) {

            cout << temp.front() << " ";

            temp.pop();

        }

        cout << endl;

    }

    bool empty() {

        if (q.empty())

            return true;

        else

            return false;

    }

};

int main() {

    StackOneQueue st;

    int choice, val;

    bool running = true;

    while (running) {

        cout << "\n1.Push\n2.Pop\n3.Top\n4.Display\n5.Check Empty\nAny other number to Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter value: ";

            cin >> val;

            st.push(val);

        } else if (choice == 2) {

            st.pop();

        } else if (choice == 3) {

            st.top();

        } else if (choice == 4) {

            st.display();

        } else if (choice == 5) {

            if (st.empty())

                cout << "Stack is empty" << endl;

            else

                cout << "Stack is not empty" << endl;

        } else {

            cout << "Exit" << endl;

            running = false;

        }

    }

    return 0;

}