Hiral Gupta(1024030818)

DSA ASS-4

Q1

#include <iostream>

using namespace std;

#define SIZE 5

int queue[SIZE];

int front = -1, rear = -1;

void enqueue(int value) {

    if (rear == SIZE - 1) {

        cout << "Queue is full\n";

        return;

    }

    if (front == -1) front = 0;

    rear++;

    queue[rear] = value;

    cout << value << " enqueued\n";

}

void dequeue() {

    if (front == -1 || front > rear) {

        cout << "Queue is empty\n";

        return;

    }

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

    front++;

    if (front > rear) front = rear = -1;

}

void peek() {

    if (front == -1)

        cout << "Queue is empty\n";

    else

        cout << "Front element: " << queue[front] << endl;

}

void display() {

    if (front == -1)

        cout << "Queue is empty\n";

    else {

        cout << "Queue: ";

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

            cout << queue[i] << " ";

        cout << endl;

    }

}

void isEmpty() {

    if (front == -1)

        cout << "Queue is empty\n";

    else

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

}

void isFull() {

    if (rear == SIZE - 1)

        cout << "Queue is full\n";

    else

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

}

int main() {

    int choice, value;

    do {

        cout << "\n1.Enqueue  2.Dequeue  3.Peek  4.Display  5.isEmpty  6.isFull  0.Exit\n";

        cout << "Enter choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter value: ";

                cin >> value;

                enqueue(value);

                break;

            case 2: dequeue(); break;

            case 3: peek(); break;

            case 4: display(); break;

            case 5: isEmpty(); break;

            case 6: isFull(); break;

            case 0: cout << "Exiting...\n"; break;

            default: cout << "Invalid choice\n";

        }

    } while (choice != 0);

    return 0;

}

Q2

#include <iostream>

using namespace std;

#define SIZE 5

int queue[SIZE];

int front = -1, rear = -1;

bool isFull() {

    return (front == (rear + 1) % SIZE);

}

bool isEmpty() {

    return (front == -1);

}

void enqueue(int value) {

    if (isFull()) {

        cout << "Queue is full\n";

        return;

    }

    if (isEmpty()) {

        front = rear = 0;

    } else {

        rear = (rear + 1) % SIZE;

    }

    queue[rear] = value;

    cout << value << " enqueued\n";

}

void dequeue() {

    if (isEmpty()) {

        cout << "Queue is empty\n";

        return;

    }

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

    if (front == rear) {

        front = rear = -1;

    } else {

        front = (front + 1) % SIZE;

    }

}

void peek() {

    if (isEmpty())

        cout << "Queue is empty\n";

    else

        cout << "Front element: " << queue[front] << endl;

}

void display() {

    if (isEmpty()) {

        cout << "Queue is empty\n";

        return;

    }

    cout << "Queue: ";

    int i = front;

    while (true) {

        cout << queue[i] << " ";

        if (i == rear)

            break;

        i = (i + 1) % SIZE;

    }

    cout << endl;

}

int main() {

    int choice, value;

    do {

        cout << "\n1.Enqueue  2.Dequeue  3.Peek  4.Display  5.isEmpty  6.isFull  0.Exit\n";

        cout << "Enter choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter value: ";

                cin >> value;

                enqueue(value);

                break;

            case 2: dequeue(); break;

            case 3: peek(); break;

            case 4: display(); break;

            case 5:

                cout << (isEmpty() ? "Queue is empty\n" : "Queue is not empty\n");

                break;

            case 6:

                cout << (isFull() ? "Queue is full\n" : "Queue is not full\n");

                break;

            case 0: cout << "Exiting...\n"; break;

            default: cout << "Invalid choice\n";

        }

    } while (choice != 0);

    return 0;

}

Q3

#include <iostream>

#include <queue>

using namespace std;

void interleaveQueue(queue<int>& q) {

    int n = q.size();

    if (n % 2 != 0) {

        cout << "Queue has odd number of elements. Cannot interleave.\n";

        return;

    }

    queue<int> firstHalf;

    int halfSize = n / 2;

    // Step 1: Put first half into a new queue

    for (int i = 0; i < halfSize; i++) {

        firstHalf.push(q.front());

        q.pop();

    }

    // Step 2: Interleave elements

    while (!firstHalf.empty()) {

        q.push(firstHalf.front());

        firstHalf.pop();

        q.push(q.front());

        q.pop();

    }

}

int main() {

    queue<int> q;

    int n, num;

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

    cin >> n;

    cout << "Enter " << n << " elements:\n";

    for (int i = 0; i < n; i++) {

        cin >> num;

        q.push(num);

    }

    interleaveQueue(q);

    cout << "Interleaved Queue:\n";

    while (!q.empty()) {

        cout << q.front() << " ";

        q.pop();

    }

    cout << endl;

    return 0;

}

#include <iostream>

#include <queue>

using namespace std;

void firstNonRepeating(string str) {

    int freq[26] = {0}; // to count character frequencies

    queue<char> q;

    for (int i = 0; i < str.length(); i++) {

        char ch = str[i];

        freq[ch - 'a']++; // update frequency

        q.push(ch);       // push into queue

        // remove characters from front if they are repeating

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

            q.pop();

        }

        if (q.empty())

            cout << "-1 ";

        else

            cout << q.front() << " ";

    }

}

int main() {

    string input;

    cout << "Enter characters (no spaces, like 'aabc'): ";

    cin >> input;

    cout << "First non-repeating characters:\n";

    firstNonRepeating(input);

    return 0;

}

Q5

#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 is empty\n";

            return;

        }

        q1.pop();

    }

    void top() {

        if (q1.empty()) {

            cout << "Stack is empty\n";

        } else {

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

        }

    }

    void display() {

        if (q1.empty()) {

            cout << "Stack is empty\n";

            return;

        }

        cout << "Stack elements: ";

        queue<int> temp = q1;

        while (!temp.empty()) {

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

            temp.pop();

        }

        cout << endl;

    }

};

int main() {

    Stack s;

    int choice, val;

    do {

        cout << "\n1.Push  2.Pop  3.Top  4.Display  0.Exit\n";

        cout << "Enter choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter value: ";

            cin >> val;

            s.push(val);

        } else if (choice == 2) {

            s.pop();

        } else if (choice == 3) {

            s.top();

        } else if (choice == 4) {

            s.display();

        }

    } while (choice != 0);

    return 0;

}