Date : 22.03.2025
Lakshana Baskaran
24011103026

LAB - 7
Queue ADT - Array & Singly Linked List

QUESTION 1:

A. Write a separate C++ menu-driven program to implement Queue ADT using an integer array of size 5. Maintain proper boundary conditions and follow good coding practices. The Queue ADT has the following operations:

1. Enqueue
2. Dequeue
3. Peek
4. Exit

SOURCE CODE:

```
//Queue ADT using array implementation with array having size 5
#include <iostream>
#include <stdbool.h>
using namespace std;
#define SIZE 5

class queue {
    private:
        int arr[SIZE];
        int rear;
    public:
        void enqueue(int);
        int dequeue();
        int peek();
        void print();
        bool isfull();
        bool isempty();


        queue() {
            rear = -1;
        }
};
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

```
//adds element to back
void queue::enqueue(int x) {
    if (isfull()) {
        printf("  OverFlowError: Queue is Full\n");
        return;
    }

    arr[rear+1] = x;
    rear++;
}

//deletes front
int queue::dequeue() {
    if (isempty()) {
        printf("  UnderFlowError: Queue is Empty\n");
        return 0;
    }

    int del = arr[0];
    if (rear == 0) {
        rear = -1;
    }
    else {
        for (int i = 0; i < rear; ++i) {
            arr[i] = arr[i+1];
        }
        rear--;
    }

    return del;
}

//rturns the front
int queue::peek() {
    if (isempty()) {
        return 0;
    }
    return arr[0];
}

//displays the queue
void queue::print() {
```

```
    for (int i = 0; i < rear+1; ++i) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

//checks if q is full
bool queue::isfull() {
    return rear == SIZE-1;
}

//checks if q is empty
bool queue::isempty() {
    return rear == -1;
}

int main() {
    queue Q;
    int x, choice = 0;

    printf("MENU\n1 - Enqueue\n2 - Dequeue\n3 - Peek\n4 - Exit\n");
    printf("Zero is not allowed in the queue. If the UnderFlowError
occurs zero will be returned.\n");
    while (true) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter Element to be enqueued: ");
                scanf("%d", &x);
                Q.enqueue(x);
                break;

            case 2:
                cout << Q.dequeue() << endl;
                break;

            case 3:
                cout << "front: " << Q.peek() << endl;
                break;

            case 4:
                printf("Exiting...\n");
```

```
                return 0;
                break;

            default:
                printf("\nInvalid choice. Enter again.\n");
                break;
        }
        if (!Q.isempty()) {
            printf("\tQueue = ");
            Q.print();
        }
    }
}
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-8$ g++ -o out q_array.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-8$ ./out
MENU
1 - Enqueue
2 - Dequeue
3 - Peek
4 - Exit
Zero is not allowed in the queue. If the UnderFlowError occurs zero will be returned.

Enter your choice: 1
Enter Element to be enqueued: 2
        Queue = 2

Enter your choice: 1
Enter Element to be enqueued: 3
        Queue = 2 3

Enter your choice: 3
front: 2
        Queue = 2 3

Enter your choice: 2
2
        Queue = 3

Enter your choice: 2
3

Enter your choice: 2
  UnderFlowError: Queue is Empty
0

Enter your choice: 6

Invalid choice. Enter again.

Enter your choice: 4
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-8$
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

QUESTION 2:

Write a separate C++ menu-driven program to implement Circular Queue ADT using an integer array of size 5. Maintain proper boundary conditions and follow good coding practices. The Circular Queue ADT has the following operations:

1. Enqueue
2. Dequeue
3. Peek
4. Exit

SOURCE CODE:

```
//Queue ADT using array implementation with array having size 5
#include <iostream>
#include <stdbool.h>
using namespace std;
#define SIZE 5

class queue {
    private:
        int arr[SIZE];
        int front;
        int rear;
    public:
        void enqueue(int);
        int dequeue();
        int peek();
        void print();
        bool isfull();
        bool isempty();


        queue() {
            front = -1;
            rear = -1;
        }
};

//inserts element of at the end of the queue
void queue::enqueue(int x) {
    if (isfull()) {
        printf("  OverFlowError: Queue is Full\n");
        return;
```

```
    }
    if (isempty()) {
        arr[front+1] = x;
        front++;
        rear++;
        return;
    }

    if (front < rear) {
        if (rear == SIZE-1) {
            arr[0] = x;
            rear = 0;
        }
        else {
            arr[rear+1] = x;
            rear++;
        }
    }

    else if (front == rear) {
        if (rear == SIZE-1) {
            arr[0] = x;
            rear = 0;
        }
        else {
            arr[rear+1] = x;
            rear++;
        }
    }

    else if (front > rear) {
        arr[rear+1] = x;
        rear++;
    }

    else {
        return;
    }

}

//Deletes the element at the beginning
int queue::dequeue() {
```

```
    if (isempty()) {
        printf("  UnderFlowError: Queue is Empty\n");
        return 0;
    }

    int del = arr[front];
    arr[front] = 0;

    if (front == rear) {
        front = -1;
        rear = -1;
    }

    else if (front < rear) {
        front++;
    }

    else if (front > rear) {
        if (front-1 == rear) {
            if (front == SIZE-1) {
                front = 0;
            }
            else {
                front++;
            }
        }
        else {
            if (front == SIZE-1) {
                front = 0;
            }
            else {
                front++;
            }
        }
    }

    return del;
}

//returns the front of the queue
int queue::peek() {
    if (isempty()) {
        return 0;
```

```
    }
    return arr[front];
}

//Displays the queue
void queue::print() {
    if (front < rear) {
        for (int i = front; i < rear+1; ++i) {
           printf("%d ", arr[i]);
        }
    }
    else if (front == rear && front != -1) {
        printf("%d", arr[front]);
    }

    else if (front > rear) {
        for (int i = front; i < SIZE; ++i) {
            printf("%d ", arr[i]);
        }
        for (int i = 0; i < rear+1; ++i) {
            printf("%d ", arr[i]);
        }
    }

    printf("\n");
}

bool queue::isfull() {
    if (rear+1 == front)
        return true;
    if (front == 0 && rear == SIZE-1)
        return true;
    return false;
}

bool queue::isempty() {
    return rear == -1;
}


int main() {
    queue Q;
    int x, choice = 0;
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

```
    printf("MENU\n1 - Enqueue\n2 - Dequeue\n3 - Peek\n4 - Exit\n");
    printf("Zero is not allowed in the queue. If the UnderFlowError
occurs zero will be returned.\n");
    while (true) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter Element to be enqueued: ");
                scanf("%d", &x);
                Q.enqueue(x);
                break;

            case 2:
                cout << Q.dequeue() << endl;
                break;

            case 3:
                cout << "front: " << Q.peek() << endl;
                break;

            case 4:
                printf("Exiting...\n");
                return 0;
                break;

            default:
                printf("\nInvalid choice. Enter again.\n");
                break;
        }
        if (!Q.isempty()) {
            printf("\tQueue = ");
            Q.print();
        }
    }
}
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

SOURCE CODE:

```
lemon@jupiter:~/workspace/college/DSA/Lab-8$ g++ -o out cq_array.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-8$ ./out
MENU
1 - Enqueue
2 - Dequeue
3 - Peek
4 - Exit
Zero is not allowed in the queue. If the UnderFlowError occurs zero will be returned.

Enter your choice: 1
Enter Element to be enqueued: 2
        Queue = 2

Enter your choice: 1
Enter Element to be enqueued: 3
        Queue = 2 3

Enter your choice: 3
front: 2
        Queue = 2 3

Enter your choice: 2
2
        Queue = 3

Enter your choice: 2
3

Enter your choice: 2
  UnderFlowError: Queue is Empty
0

Enter your choice: 5

Invalid choice. Enter again.

Enter your choice: 4
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-8$
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

QUESTION 3:

Write a separate C++ menu-driven program to implement Queue ADT using an integer-linked list. Maintain proper boundary conditions and follow good coding practices. The Queue ADT has the following operations:

1. Enqueue
2. Dequeue
3. Peek
4. Exit

SOURCE CODE:

```
//Implementation of stack ADT using singly linked list
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <stdbool.h>
using namespace std;

class node {
    private:
        int data;
        struct node *next;
        struct node *head;

    public:
        node() {
            head = NULL;
        }

        void enqueue(int);
        int dequeue();
        int peek();
        bool isempty();

        void print();

};

//adds element to end
void node::enqueue(int x) {
    struct node* newnode = new node;
    newnode -> data = x;
```

```
    newnode -> next = NULL;

    if (head == NULL) {
        head = newnode;
        return;
    }

    struct node* temp = head;
    for (; temp -> next != NULL; temp = temp -> next) {
    }
    temp -> next = newnode;

}

//deletes the front
int node::dequeue() {
    if (head == NULL) {
        printf("UnderFlow Error: Stack is Empty.\n");
        return 0;
    }

    int elem;
    elem = head -> data;
    head = head -> next;
    return elem;
}

//returns the front
int node::peek() {
    return head -> data;
}

//displays the queue
void node::print() {
    struct node *temp = head;
    if (temp == NULL) {
        printf("head = NULL\n");
        return;
    }

    //printf("head -> ");
    for (; temp -> next != NULL; temp = temp -> next) {
        cout << temp -> data << " ";
```

```
    }
    cout << temp -> data << "\n";
}

//checks if queue is empty
bool node::isempty() {
    return (head == NULL);
}


int main() {
    node Q;
    int x, choice = 0;

    printf("MENU\n1 - Enqueue\n2 - Dequeue\n3 - Peek\n4 - Exit\n");
    printf("Zero is not allowed in the queue. If the UnderFlowError
occurs zero will be returned.\n");
    while (true) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter Element to be enqueued: ");
                scanf("%d", &x);
                Q.enqueue(x);
                break;

            case 2:
                cout << Q.dequeue() << endl;
                break;

            case 3:
                cout << "front: " << Q.peek() << endl;
                break;

            case 4:
                printf("Exiting...\n");
                return 0;
                break;

            default:
                printf("\nInvalid choice. Enter again.\n");
                break;
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

```
        }
        if (!Q.isempty()) {
            printf("\tQueue = ");
            Q.print();
        }
    }
}
```

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-8$ g++ -o out q_sll.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-8$ ./out
MENU
1 - Enqueue
2 - Dequeue
3 - Peek
4 - Exit
Zero is not allowed in the queue. If the UnderFlowError occurs zero will be returned.

Enter your choice: 1
Enter Element to be enqueued: 3
        Queue = 3

Enter your choice: 1
Enter Element to be enqueued: 4
        Queue = 3 4

Enter your choice: 3
front: 3
        Queue = 3 4

Enter your choice: 2
3
        Queue = 4

Enter your choice: 2
4

Enter your choice: 2
UnderFlow Error: Stack is Empty.
0

Enter your choice: 6

Invalid choice. Enter again.

Enter your choice: 4
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-8$
```

Date : 22.03.2025
Lakshana Baskaran
24011103026

QUESTION 4:

Take a string from the user that consists of the '+' symbol. Process the string such that the final string does not include the '+' symbol and the immediate left non-'+' symbol. Select and choose the optimal ADT. Implement the program by including the appropriate header file.

Example:

Input: 45fgd+++abt+c

Output: 45ac

SOURCE CODE:

```
//prcessing the string such that '+' and the character to the left of
'+' are eleminated
#include "char_stack.h"

//function declarations
string processing(string);

//45fgd+++ab+c
int main() {
    string input;
    string output;

    cout << "Enter the string: ";
    cin >> input;

    output = processing(input);
    cout << "String after proccessing: " << output << endl;
}

//the function that processes the string according ot the given
conditions
string processing(string str) {
    node stk1;
    node stk2;
    char c;
    string out;
    int i;

    for (char ch : str) {
        stk1.push(ch);
    }
```

```
    while (!stk1.empty()) {
        i = 0;
        if (stk1.peek() == '+') {
            while (stk1.peek() == '+' && !stk1.empty()) {
                stk1.pop();
                i++;
            }
            int j = 0;
            while (j < i) {
                if (stk1.empty()) {
                    break;
                }
                stk1.pop();
                j++;
            }
        }

        else {
            c = stk1.pop();
            stk2.push(c);
        }
    }


    while (!stk2.empty()) {
        c = stk2.pop();
        out = out + c;
    }

    return out;
}
```

SOURCE CODE:

```
lemon@jupiter:~/workspace/college/DSA/Lab-8$ g++ -o out eliminating_plus.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-8$ ./out
Enter the string: 45fgd+++ab+c
String after proccessing: 45ac
lemon@jupiter:~/workspace/college/DSA/Lab-8$
```

—