**EXPERIMENT-2**

**1. STACK IMPLEMENTATION USING ARRAY**

**Code:**

#include <iostream>

using namespace std;

struct stack{

    int size;

    int top;

    int \*s;

};

void push(stack \*st){

    if(st->top == st->size-1){

        cout<<"Stack overflow."<<endl;

    }

    else{

        int data;

        cout<<"Enter data to push : ";

        cin>>data;

        st->top++;

        st->s[st->top] = data;

        cout<<"Pushed "<<data<<" successfully."<<endl;

    }

}

int pop(stack \*st){

    int pop\_data = -1;

    if(st->top == -1){

        cout<<"Stack underflow."<<endl;

    }

    else{

        pop\_data = st->s[st->top];

        st->top--;

    }

    return pop\_data;

}

void display(stack \*st){

    cout<<" - "<<endl;

    for(int i = st->top; i>=0; i--){

        cout<<"|"<<st->s[i]<<"|"<<endl;

    }

    cout<<" - "<<endl;

}

int main(){

    struct stack s1;

    cout<<"Enter size of stack: ";

    cin>>s1.size;

    s1.s = new int[s1.size];

    s1.top = -1;

    push(&s1);

    push(&s1);

    push(&s1);

    push(&s1);

    display(&s1);

    cout<<"Popped: "<<pop(&s1)<<endl;

    cout<<"Popped: "<<pop(&s1)<<endl;

    display(&s1);

    return 0;

}

**2. QUEUE IMPLEMENTATION USING ARRAY**

**Code:**

#include <iostream>

using namespace std;

class Queue

{

private:

    int size;

    int front;

    int rear;

    int \*Q;

public:

    Queue()

    {

        size = 10;

        front = rear = -1;

        Q = new int[size];

    }

    Queue(int size)

    {

*this*->size = size;

        front = rear = -1;

        Q = new int[*this*->size];

    }

    void enqueue();

    int dequeue();

    void display();

};

void Queue::enqueue()

{

    if (rear == size - 1)

    {

        cout << "Queue is full.\n";

    }

    else

    {

        int data;

        cout << "Enter the data to add to the queue : ";

        cin >> data;

        rear++;

        Q[rear] = data;

        cout << data << " added.\n\n";

    }

}

int Queue::dequeue()

{

    int x = -1;

    if (front == rear)

    {

        cout << "Queue is empty.\n";

    }

    else

    {

        front++;

        x = Q[front];

        cout << x << " removed.\n\n";

    }

    return x;

}

void Queue::display()

{

    for (int i = front + 1; i <= rear; i++)

    {

        cout << "|" << Q[i] << "|\n";

    }

}

int main()

{

    Queue q[5];

    q->enqueue();

    q->enqueue();

    q->enqueue();

    q->enqueue();

    q->dequeue();

    q->display();

    return 0;

}