2. Write a C++ programs to implement the following using an array.

a) Stack ADT

b) Queue ADT

a) STACK ADT:-

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

class Stack {
private:
   int top;
   int arr[5]; // Fixed-size stack (can be modified)
   int maxSize;

public:
   Stack() {
      top = -1;        // Stack is initially empty
      maxSize = 5;     // Fixed maximum size
   }

   bool isFull() {
      return top == maxSize - 1;
   }

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

   void push(int x) {
      if (isFull()) {
         cout << "Stack Overflow! Cannot push " << x << endl;
      } else {
```

```
            arr[++top] = x;
            cout << x << " pushed to stack." << endl;
        }
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow! Cannot pop." << endl;
        } else {
            cout << arr[top--] << " popped from stack." << endl;
        }
    }

    void peek() {
        if (isEmpty()) {
            cout << "Stack is empty, nothing to peek." << endl;
        } else {
            cout << "Top element is " << arr[top] << endl;
        }
    }
};

int main() {
    Stack stack;

    stack.push(10);
    stack.push(20);
    stack.push(30);
    stack.peek(); // Should return 30
    stack.pop();  // Should pop 30
    stack.peek(); // Should return 20
```

```
    return 0;
}
```

B) QUEUE ADT:-

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

class Queue {
private:
    int front;
    int rear;
    int arr[5]; // Fixed-size queue (can be modified)
    int maxSize;

public:
    Queue() {
        front = -1;
        rear = -1;
        maxSize = 5;
    }

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

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

```
void enqueue(int x) {
    if (isFull()) {
        cout << "Queue Overflow! Cannot enqueue " << x << endl;
    } else {
        if (front == -1) // Initial insertion
            front = 0;
        arr[++rear] = x;
        cout << x << " enqueued to queue." << endl;
    }
}

void dequeue() {
    if (isEmpty()) {
        cout << "Queue Underflow! Cannot dequeue." << endl;
    } else {
        cout << arr[front] << " dequeued from queue." << endl;
        if (front == rear) { // Reset queue if last element is dequeued
            front = rear = -1;
        } else {
            front++;
        }
    }
}

void peek() {
    if (isEmpty()) {
        cout << "Queue is empty, nothing to peek." << endl;
    } else {
        cout << "Front element is " << arr[front] << endl;
    }
}
```

```
};

int main() {
    Queue queue;

    queue.enqueue(10);
    queue.enqueue(20);
    queue.enqueue(30);
    queue.peek();  // Should return 10
    queue.dequeue(); // Should dequeue 10
    queue.peek();  // Should return 20

    return 0;
}
```