**Task 2**

1. In the given queue, front and rear has been initialized with 0. Insert func ion of the queue has been given as:

**Ans:**

void insert(int x) {

if (rear + 1 == size) {

cout << "Queue overflow" << endl;

return;

}

arr[++rear] = x;

}

1. In the circular queue, front and rear has been initialized with -1. Insert function of the

queue has been given as:

**Ans:**

void insert(int x) {

if ((rear + 1) % size == front) {

cout << "Queue overflow" << endl;

return;

}

arr[(++rear) % size] = x;

}

1. In the display function, following code has been written:

**Ans:**

void display()

{

int beg = 0;

while (beg <= top)

cout << stack[beg++] << endl;

}

void display()

{

int t = top;

while (t >= 0)

cout << stack[t--] << endl;

}

The second version would be preferred because it iterates through the stack elements in the correct order (from top to bottom), which is more intuitive and commonly expected in a stack data structure.

1. In a stack-based template, pop function is written; find out the possible problem that

may occur in this code.

**Ans:**

some potential problems in the provided pop() function:

1. Incorrect Order of Operations: The code tries to access top->data before checking for top==NULL. If the stack is empty (top is NULL), attempting to access top->data will lead to a segmentation fault or crash.
2. Memory Leak: The code calls delete top without first assigning top->next to top. This means the node after the removed top node is still allocated in memory but no longer accessible, causing a memory leak.

pop() {

if (top == NULL) {

cout << "Stack underflow" << endl;

return NULL; // Or throw an exception

}

t x = top->data;

Node\* temp = top;

top = top->next; // Update top pointer first

delete temp; // Then delete the previous top node

return x;

}