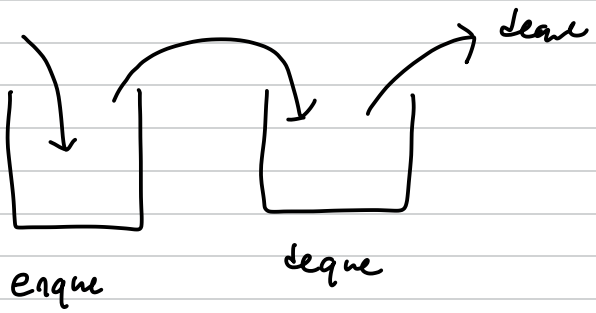
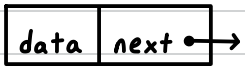

deque
enque
deque

```c
struct Node {
    int data;
    struct Node* next;
};
```



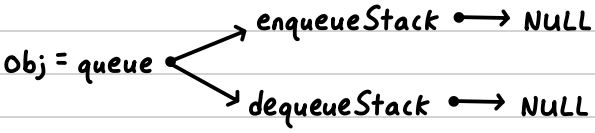

```c
typedef struct {
    struct Node* enqueueStack;
    struct Node* dequeueStack;
} MyQueue;
```

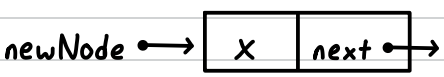
enqueueStack ⟶

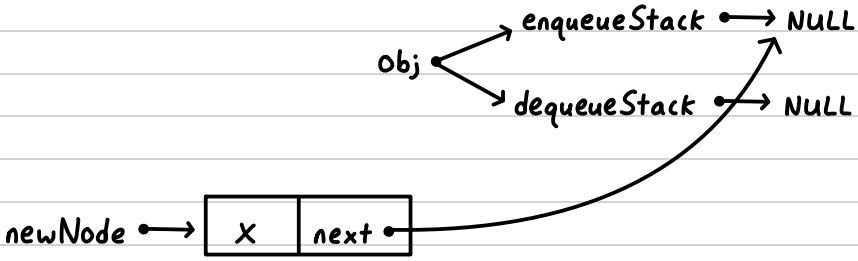
dequeueStack ⟶

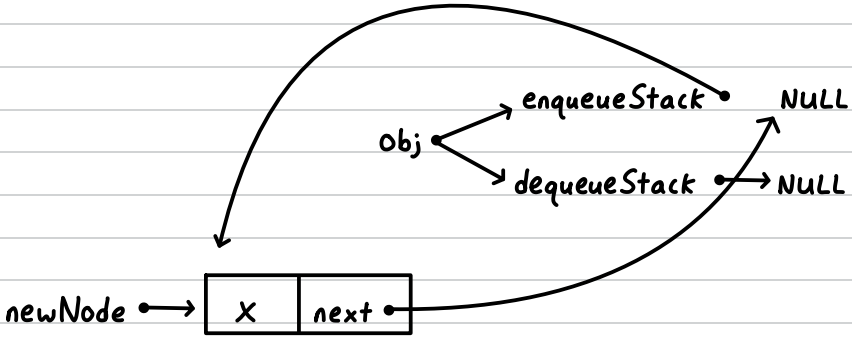
```c
MyQueue* myQueueCreate() {
    MyQueue* queue = (MyQueue*)malloc(sizeof(MyQueue));
    queue->enqueueStack = NULL;
    queue->dequeueStack = NULL;

    return queue;
}
```

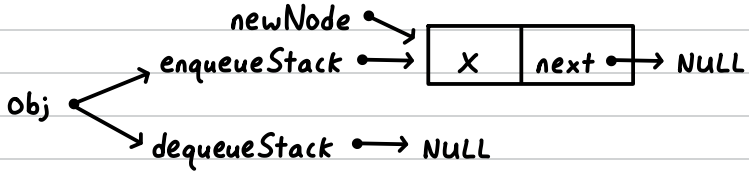

```c
void myQueuePush(MyQueue* obj, int x) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = x;
    newNode->next = obj->enqueueStack;
    obj->enqueueStack = newNode;
}
```



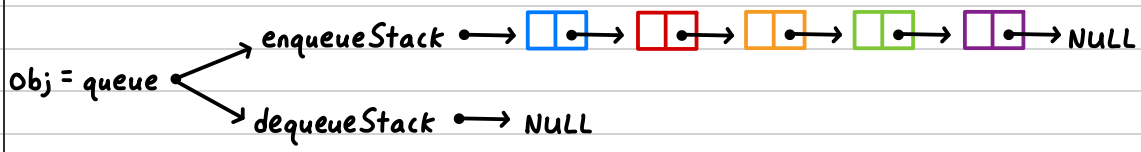


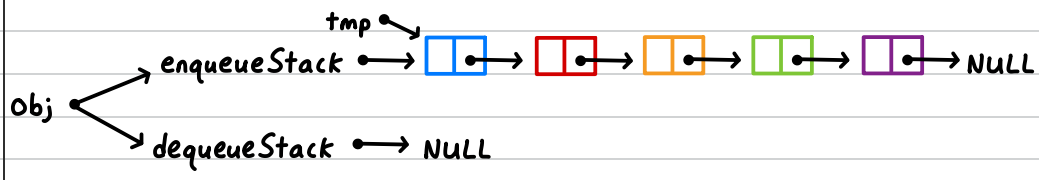


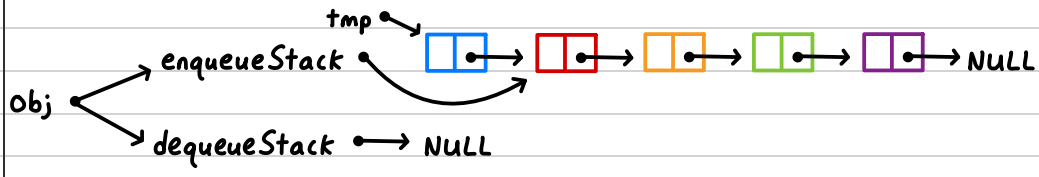


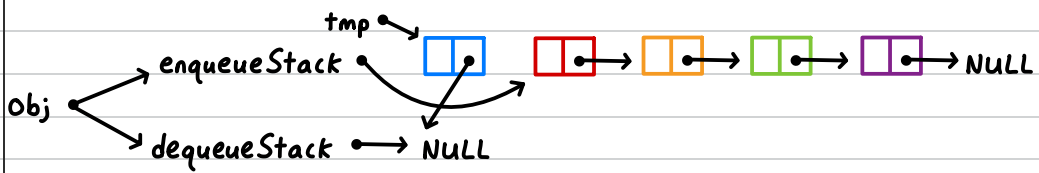


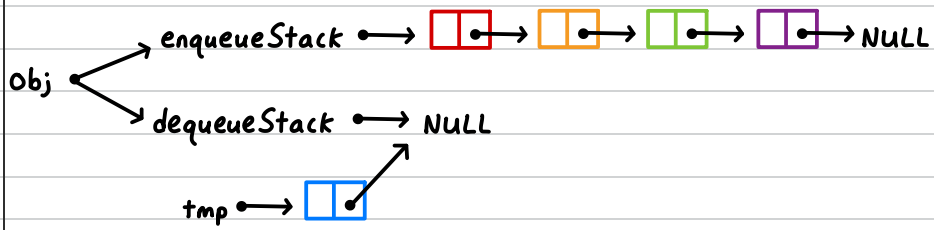
```c
int myQueuePop(MyQueue* obj) {
    if (obj->dequeueStack == NULL) {
        while (obj->enqueueStack != NULL) {
            struct Node* temp = obj->enqueueStack;
            obj->enqueueStack = (obj->enqueueStack)->next;
            temp->next = obj->dequeueStack;
            obj->dequeueStack = temp;
        }
    }
    if (obj->dequeueStack == NULL) {
        printf("underflow. no elements in enqueue and dequeue stacks.\n");
        return -1;
    }

    struct Node* temp = obj->dequeueStack;
    int popped = temp->data;
    obj->dequeueStack = (obj->dequeueStack)->next;
    free(temp);
    return popped;
}
```

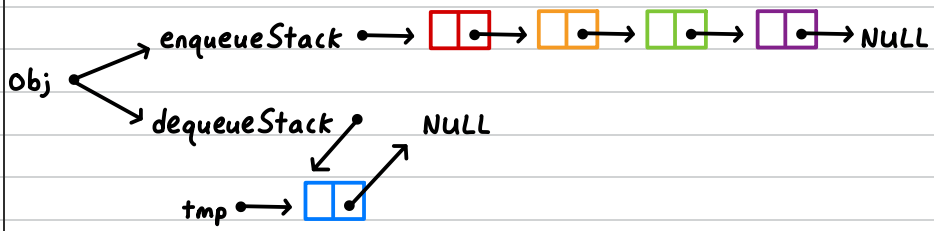

Obj = queue
enqueueStack
NULL
dequeueStack
NULL

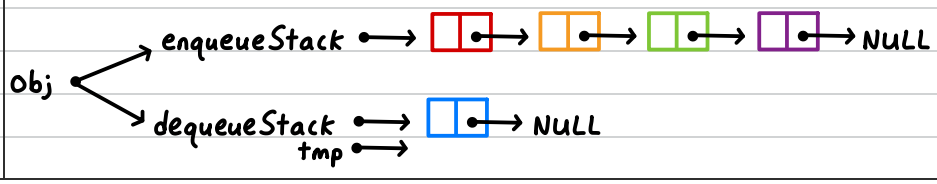

tmp
Obj
enqueueStack
NULL
dequeueStack
NULL

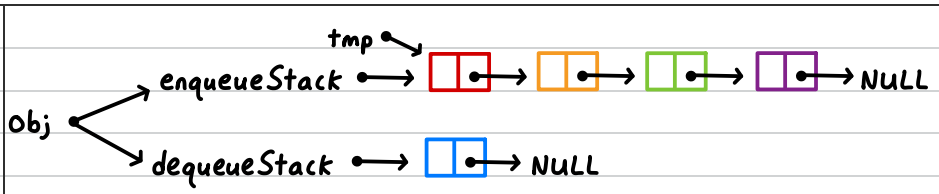

tmp
Obj
enqueueStack
NULL
dequeueStack
NULL

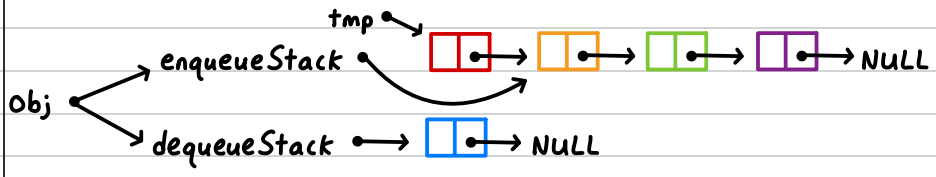

tmp
Obj
enqueueStack
NULL
dequeueStack
NULL

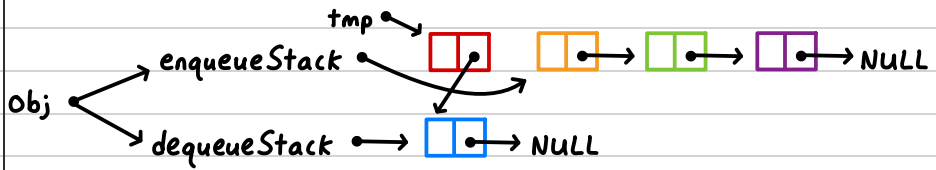

Obj
enqueueStack
NULL
dequeueStack
NULL
tmp

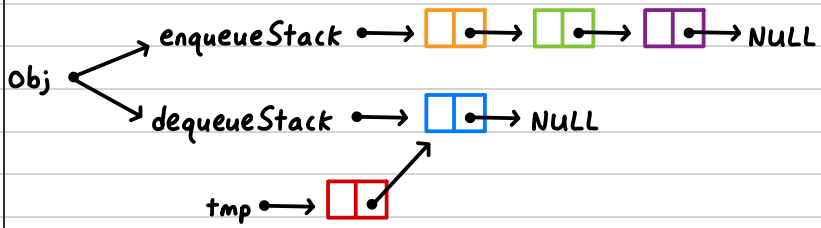

Obj
enqueueStack
NULL
dequeueStack
NULL
tmp

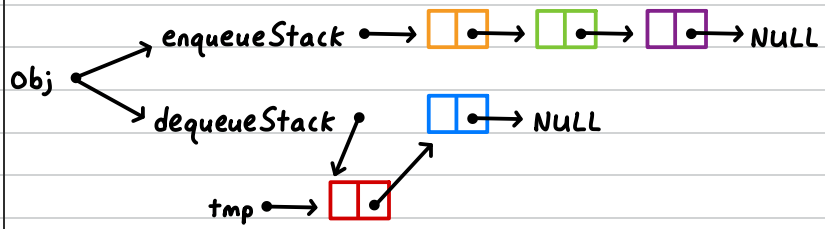

Obj
enqueueStack
NULL
dequeueStack
NULL
tmp

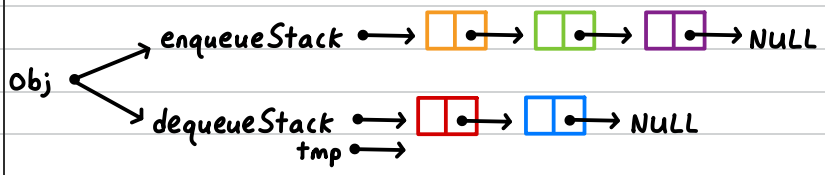






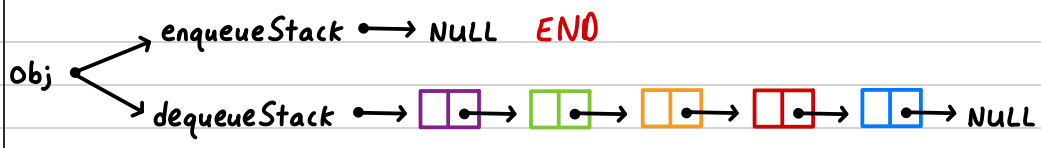


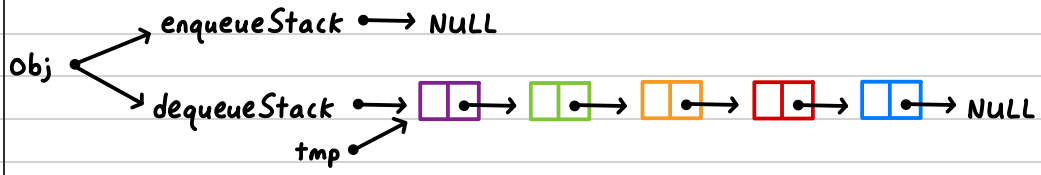


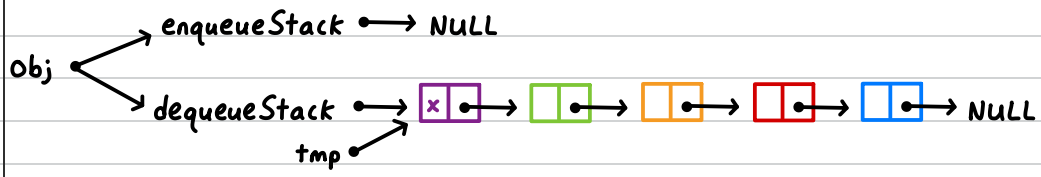


while obj → enqueueStack != NULL

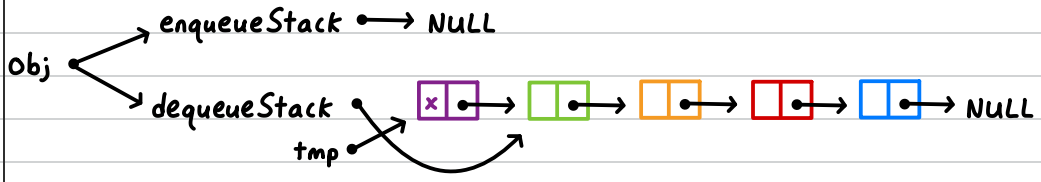
Final



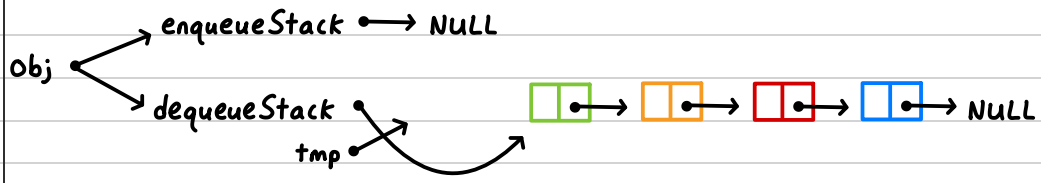




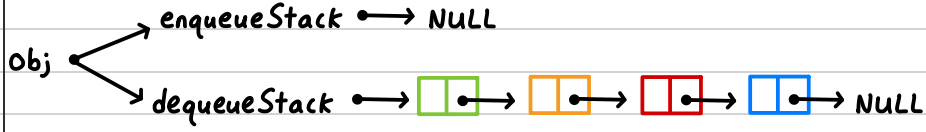
popped = x



popped = x



popped = x

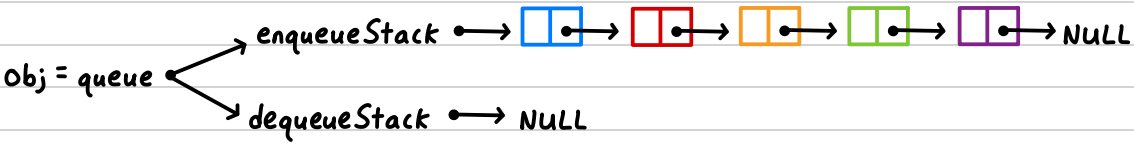


popped = x

return popped

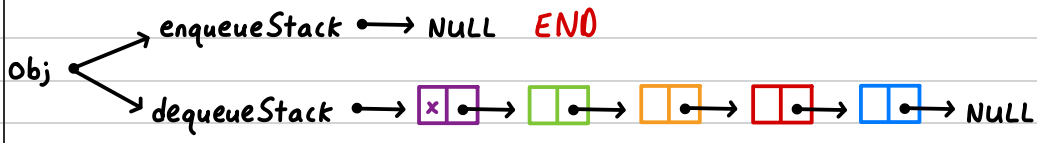
```c
int myQueuePeek(MyQueue* obj) {
    if (obj->dequeueStack == NULL) {
        while (obj->enqueueStack != NULL) {
            struct Node* temp = obj->enqueueStack;
            obj->enqueueStack = (obj->enqueueStack)->next;
            temp->next = obj->dequeueStack;
            obj->dequeueStack = temp;
        }
    }
    if (obj->dequeueStack == NULL) {
        printf("underflow. no elements in enqueue and dequeue stacks.\n");
        return -1;
    }
    return (obj->dequeueStack)->data;
}
```

same as pop, but no deletion



while obj → enqueueStack != NULL

Final

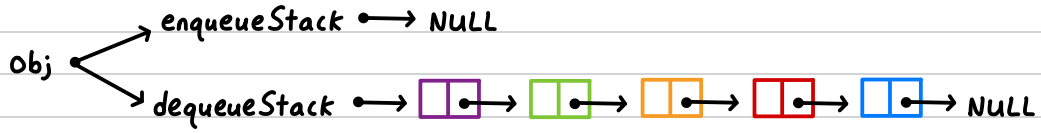


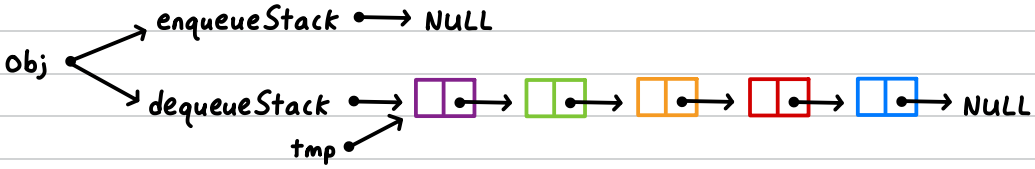
return x

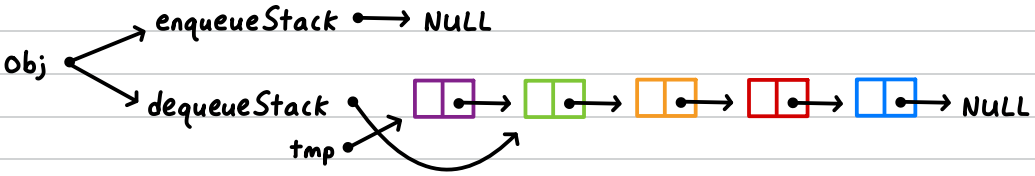
```c
bool myQueueEmpty(MyQueue* obj) {
    return obj->enqueueStack == NULL && obj->dequeueStack == NULL;
}
```

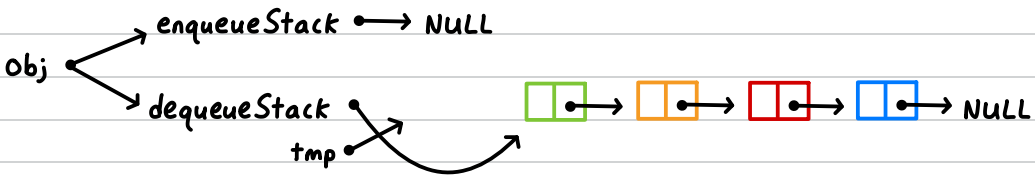
True → [enqueueStack → NULL] ∧ [dequeueStack → NULL]

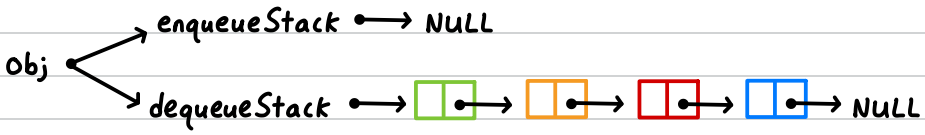
```c
void myQueueFree(MyQueue* obj) {
    while (obj->enqueueStack != NULL) {
        struct Node* temp = obj->enqueueStack;
        obj->enqueueStack = (obj->enqueueStack)->next;
        free(temp);
    }
    while (obj->dequeueStack != NULL) {
        struct Node* temp = obj->dequeueStack;
        obj->dequeueStack = (obj->dequeueStack)->next;
        free(temp);
    }
    free(obj);
}
```











while obj → enqueueStack != NULL
obj → dequeueStack != NULL

Final