Program 6b

Write A Program to Implement Single Link List to simulate Stack & Queue Operations.

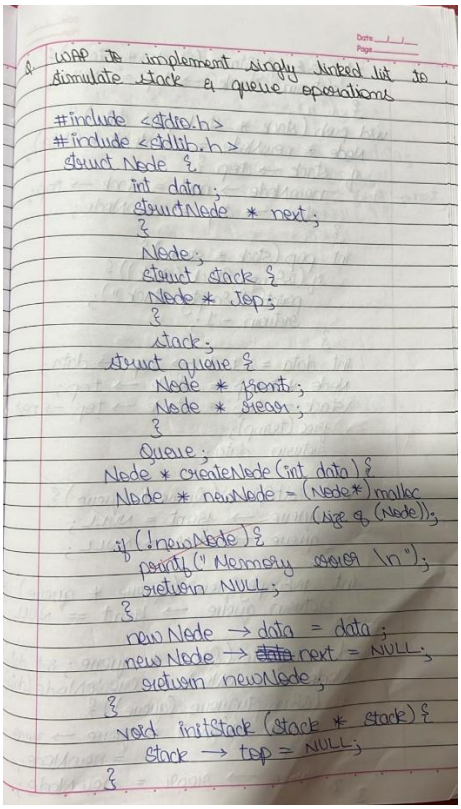
Code:

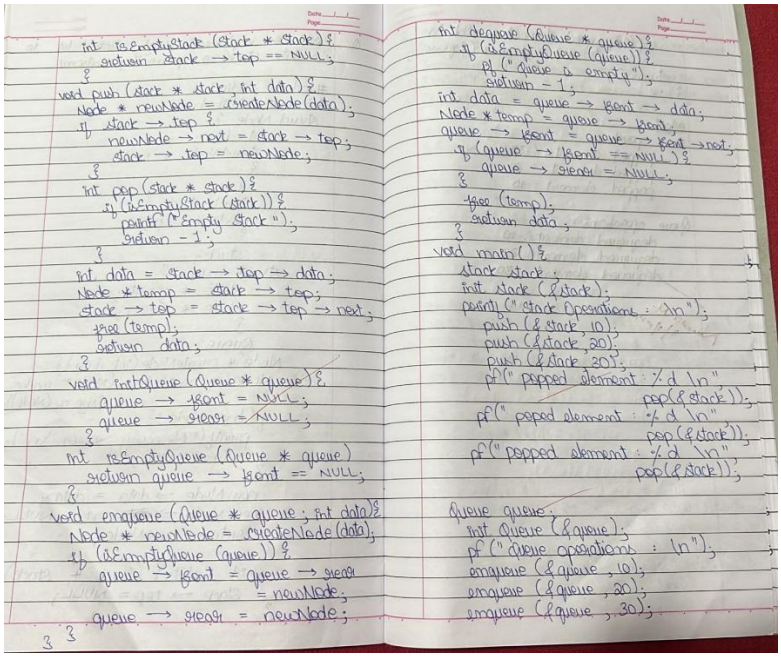
```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
   int data;
   struct Node* next;
} Node;

Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   if (!newNode) {
      printf("Memory allocation error\n");
      return NULL;
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

void push(Node** top, int data) {
   Node* newNode = createNode(data);
   if (!newNode) return;
   newNode->next = *top;
   *top = newNode;
   printf("%d pushed to stack\n", data);
}

int pop(Node** top) {
   if (*top == NULL) {
      printf("Stack Underflow\n");
      return -1;
   }
   Node* temp = *top;
   int poppedData = temp->data;
   *top = temp->next;
   free(temp);
   printf("%d popped from stack\n", poppedData);
   return poppedData;
}

void displayStack(Node* top) {
   if (top == NULL) {
      printf("Stack is empty\n");
      return;
```

```
    }
    printf("Stack: ");
    Node* temp = top;
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = createNode(data);
    if (!newNode) return;
    if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
    printf("%d enqueued to queue\n", data);
}

int dequeue(Node** front, Node** rear) {
    if (*front == NULL) {
        printf("Queue Underflow\n");
        return -1;
    }
    Node* temp = *front;
    int dequeuedData = temp->data;
    *front = temp->next;
    if (*front == NULL) {
        *rear = NULL;
    }
    free(temp);
    printf("%d dequeued from queue\n", dequeuedData);
    return dequeuedData;
}

void displayQueue(Node* front) {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue: ");
    Node* temp = front;
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
```

```
}

int main() {
    Node* stackTop = NULL;

    printf("\n--- Stack Operations ---\n");
    push(&stackTop, 10);
    push(&stackTop, 20);
    push(&stackTop, 30);
    displayStack(stackTop);
    pop(&stackTop);
    displayStack(stackTop);

    Node* queueFront = NULL;
    Node* queueRear = NULL;

    printf("\n--- Queue Operations ---\n");
    enqueue(&queueFront, &queueRear, 1);
    enqueue(&queueFront, &queueRear, 2);
    enqueue(&queueFront, &queueRear, 3);
    displayQueue(queueFront);
    dequeue(&queueFront, &queueRear);
    displayQueue(queueFront);

    return 0;
}
```

```
--- Stack Operations ---
10 pushed to stack
20 pushed to stack
30 pushed to stack
Stack: 30 -> 20 -> 10 -> NULL
30 popped from stack
Stack: 20 -> 10 -> NULL

--- Queue Operations ---
1 enqueued to queue
2 enqueued to queue
3 enqueued to queue
Queue: 1 -> 2 -> 3 -> NULL
1 dequeued from queue
Queue: 2 -> 3 -> NULL
```





```
int isEmptyStack (Stack * stack){
    return stack -> top == NULL;
}
void push (stack * stack, int data){
    Node * newNode = createNode(data);
    if stack -> top {
        newNode -> next = stack -> top;
        stack -> top = newNode;
    }
    int pop (stack * stack){
        if (isEmptyStack (stack)){
        printf ("Empty Stack");
            return -1;
    }
    int data = stack -> top -> data;
    Node *temp = stack -> top;
    stack -> top = stack -> top -> next;
    free (temp);
    return data;
    }
    void initQueue (Queue * queue){
        queue -> front = NULL;
        queue -> rear = NULL;
    }
    int isEmptyQueue (Queue * queue)
        return queue -> front == NULL;
}
void enqueue (Queue * queue, int data){
    Node * newNode = createNode (data);
    if (isEmptyQueue (queue)){
        queue -> front = queue -> rear
                        = newNode;
        queue -> rear = newNode;
}
```

```
int dequeue (Queue * queue){
    if (isEmptyQueue (queue)){
        pf ("Queue is empty");
        return -1;
    }
    int data = queue -> front -> data;
    Node *temp = queue -> front;
    queue -> front = queue -> front -> next;
    if (queue -> front == NULL){
        queue -> rear = NULL;
    }
    free (temp);
    return data;
}
void main(){
    stack stack;
    init stack (&stack);
    printf ("Stack Operations : \n");
        push (&stack, 10);
        push (&stack, 20);
        push (&stack, 30);
        pf(" popped element : %d \n"
                        pop(&stack));
    pf(" popped element : %d \n"
                        pop(&stack));
    pf(" popped element : %d \n"
                        pop(&stack));

Queue queue;
    init Queue (&queue);
    pf ("Queue operations : \n");
    enqueue (&queue, 10);
    enqueue (&queue, 20);
    enqueue (&queue, 30);
```

```
    pf ("Dequeued elements : %d \n"
                    dequeue (&queue));
    return 0;
}

output
Stack operations :
    popped element : 30
    popped element : 20
    popped element : 10

Queue operations :
    dequeued element : 10
    dequeued element : 20
    dequeued element : 30.
```