## 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
```

- 1 1 1 d	
MAY 10 MINTER	emont simply limited his to stimulate stack of
Obcastration	The state of the s
Historia Co	dianit series
# include L&	KI O N)
Smet Ned	lef
int dat	del
Charel	Nod. There.
7 Made	Strop was the same
State .	SAIRS
Nodex	lop;
3. Stock	To god may 1 of the play to the
Struck 9	uruel - mais sand a shall
Nool	e* front;
Nec	det reat;
3 8u	eut;
Node *	CreateNode fint dasa) {
No	de * newNode = [Node *) malloc (size of (Node
- II	(Incis) Node ) { printy ["Memory emorly");
1	printy ("Memory enorm");
2	relian Null;
	w Nide > dasa = dasa;
	w Node → next = Null;
Y	Corn newNode;
Void	inisack(stack + stack) {
1.4111115	fack > lop = NULL;
3	Tall the same
INI	istimply Stack (Stack # stack) &
	return Stack -> top = = NULL;
7	

	- 6
-	
S	vold push (stack + stack, int data) !
	Node + new Nede = excute Nodeldain);
	New Node -> nert = stack -> top;
	stack -> lop = new Node;
	int pop (stack + slack)
	if (ix Empty Stack (stack))
	prints ("Emply Stack ?)
	return -1;
	n to a law a data'
	int dala = Start => top > data;
	Nede * king = Stack 7 top;
	Stack - lop = Stack -> hap -> well;
	Prec Charp !
	Yelum data;
	void fait Surve ( Quene & quene) }
(C)	queue - front = NULL;
100	queue -> rear = North
	queue - year - was
1	ins istumpy Queue (Queue & queue) f
	return queue -> front == NVILI;
1	return queue - jun
	void enqueue (Queue + queue; int data)?
	void Enquene ( duene + quent , int and).
	Node * New Mode = crease Node (data);
1	if (is Empry Queue (queue))}
	queue - front = queue - rear = newlor
1	queue + rear = vew Node;
1	1 (STATE STATE STATE STATE STATE
	yavas er - yros rurr
-	