6b)WAP to Implement Single Link List to simulate Stack & Queue Opera,ons

CODE:

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

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

void push(node **head, int new_data)
{
   node *new_node = (node *)malloc(sizeof(node));
new_node->data = new_data;    new_node->next =
NULL;

   if (*head == NULL)
   {
      *head = new_node;
   }
   else
   {
      node *temp = *head;
      while (temp->next != NULL)
      {
         temp = temp->next;
      }
      temp->next = new_node;
   }
}

void pop(node **head)
{
   if (*head == NULL)
   {
      prin]("Stack is empty\n");
```

```
   }
   else
   {
      node *temp = *head;        node
*prev = NULL;

      while (temp->next != NULL)
      {
         prev = temp;
         temp = temp->next;
      }

      if (prev == NULL)
      {
         // Only one element in the list
         *head = NULL;
      }       else
      {
         prev->next = NULL;
      }

      prin]("Popped element: %d\n", temp->data);
      free(temp);
   }
}

void enqueue(node **front, int new_data)
{
   node *new_node = (node *)malloc(sizeof(node));
new_node->data = new_data;     new_node->next =
NULL;

   if (*front == NULL)
   {
      *front = new_node;
   }
   else
```

```
    {
        node *temp = *front;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = new_node;
    }
}

void dequeue(node **front)
{
    if (*front == NULL)
    {
        prin]("Queue is empty\n");
    }
    else
    {
        node *temp = *front;        *front
= temp->next;

        prin]("Dequeued element: %d\n", temp->data);        free(temp);
    }
}

void display(node *list)
{
    node *current = list;
    while (current != NULL)
    {
        prin]("%d ", current->data);
        current = current->next;
    }
    prin]("\n");
}
int main()
{
```

```
    node *stack = NULL;     node
*queue = NULL;

    // Stack opera,ons     push(&stack,
1);     push(&stack, 2);     push(&stack,
3);

    // Display the stack     prin]("Stack:
");     display(stack);

    // Pop elements from the stack
pop(&stack);     pop(&stack);
pop(&stack);

    // Queue opera,ons     enqueue(&queue, 4);
enqueue(&queue, 5);     enqueue(&queue, 6);

    // Display the queue     prin]("Queue:
");     display(queue);

    // Dequeue elements from the queue
dequeue(&queue);     dequeue(&queue);
dequeue(&queue);

    return 0;
}
```

OUTPUT: