1) WAP to Implement Single Link List to simulate Stack & Queue Operations.

```
#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)
   {
      printf("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;
        }

        printf("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)
    {
        printf("Queue is empty\n");
    }
    else
    {
        node *temp = *front;
        *front = temp->next;

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

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

```
int main()
{
    node *stack = NULL;
    node *queue = NULL;

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

    // Display the stack
    printf("Stack: ");
    display(stack);

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

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

    // Display the queue
    printf("Queue: ");
    display(queue);

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

    return 0;
}
```

```
Stack: 1 2 3
Popped element: 3
Popped element: 2
Popped element: 1
Queue: 4 5 6
Dequeued element: 4
Dequeued element: 5
Dequeued element: 6

Process returned 0 (0x0)   execution time : 2.363 s
Press any key to continue.
```