WAP to Implement Single Link List to simulate Stack & Queue Operations

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

struct node{
   int info;
   struct node * next;
};

struct node* insertintostack(struct node* top,int val){
   struct node*p;
   p=(struct node*)malloc(sizeof(struct node));
   p->info=val;

   if(top==NULL){
      p->next=NULL;
      top=p;

   }
   else{

      p->next=top;
      top=p;
   }
   return top;
}
```

```
struct node* deletefromstack(struct node* top){
   struct node* temp;

   if(top==NULL){
      return top;
   }
   else{
      temp=top;
      top=top->next;
      free(temp);
   }
   return top;

}

struct node* insertintoqueue(struct node* front,int val){
   struct node* p,*rear;
   p=(struct node*)malloc(sizeof(struct node));
   p->info=val;

   if(front==NULL){
      p->next=NULL;
      front=p;
      rear=p;

   }
   else{
      rear=front;
```

```
        while(rear->next!=NULL){
            rear=rear->next;
        }

        rear->next=p;
        p->next=NULL;

    }
    return front;
}

struct node* deletefromqueue(struct node* front){
    struct node* temp;

    if(front==NULL){
        return front;
    }
    else{
        temp=front;
        front=front->next;
        free(temp);
    }
    return front;
}

void display(struct node*start){
  struct node*temp;
```

```
 if(start==NULL){
  printf("empty\n");
 }
 else{
  temp=start;
  printf("elements are\n");
  while(temp!=NULL){
     printf("%d\n",temp->info);
     temp=temp->next;
  }
 }
}

int main(){
   struct node *top=NULL,*front=NULL;
   int choice;
   int val;
   while(1){
      printf("\n 1)Insert into stack\n 2)Delete from stack\n 3)Insert into queue\n ");
      printf("4)Delete from queue\n 5)display\n 6)exit\n");
      printf("enter choice\n");
      scanf("%d",&choice);

      switch(choice){

         case 1:
         printf("Enter value to insert\n");
         scanf("%d",&val);
```

```
top=insertintostack(top,val);
break;

case 2:
top=deletefromstack(top);
break;

case 3:
printf("Enter value to insert\n");
scanf("%d",&val);
front=insertintoqueue(front,val);
break;

case 4:
front=deletefromqueue(front);
break;

case 5:
printf("stack elements are\n");
display(top);
printf("\nqueue elements are\n");
display(front);
break;

case 6:
printf("exiting program\n");
return 0;
```

```
            default:
            return 0;
        }

    }

    return 0;
}
```

Output:

```
 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
1
Enter value to insert
2

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
1
Enter value to insert
3

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
2

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
3
```

```
Enter value to insert
4

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
4

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
5
stack elements are
elements are
2

queue elements are
empty

 1)Insert into stack
 2)Delete from stack
 3)Insert into queue
 4)Delete from queue
 5)display
 6)exit
enter choice
6
exiting program
```