Linked representation of stack

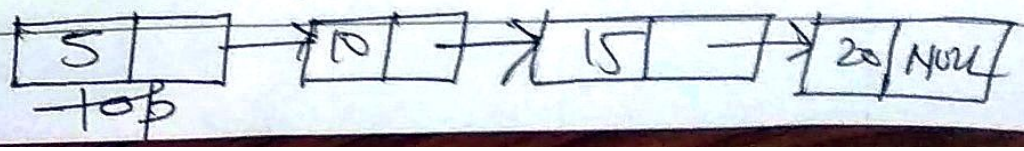
Stack can be implemented through linked list also. To implement stack using linked list we apply the stack properties on linked list. Means we insert or delete any node only from one end i.e top or end of the list.

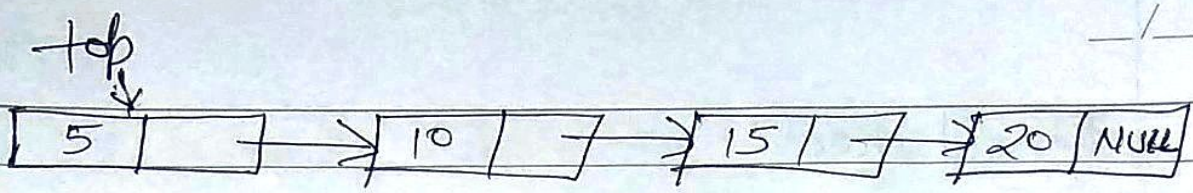
Here we take a pointer top which point to the last node of linked list.

Create stack using push() function.

```
struct node
{
    int info;
    struct node *link;
} *top = NULL;

void push()
{
    struct node *tmp;
    int m;
    tmp = (void *) malloc (sizeof (struct node));
    printf("Enter value of m for insertion");
    scanf("%d", &m);
    tmp -> info = m;
    tmp -> link = top;
    top = tmp;
}
```

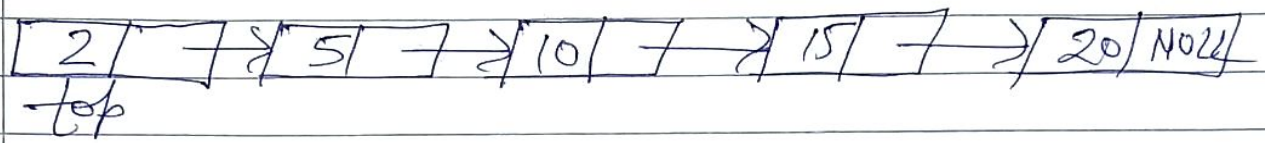




Suppose value of m = 2

```
tmp -> info = m;
tmp -> link = top;
top = tmp;
```

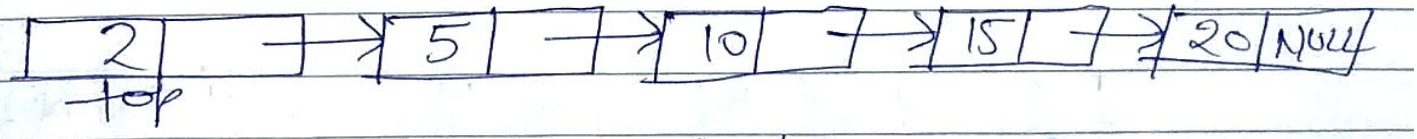
After execution above algorithm stack will be



For pushing the element on stack we will follow the insertion operation of linked list. we add the element at the start of list only.
Here top always points to the first node of linked list. Since stack implemented through linked list so we do not check overflow condition.

Pop operation on Stack :

For pop operation on stack we delete the first element of linked list because stack is last in first out structure.
Here first we check for stack empty condition then we pop the first element of stack.

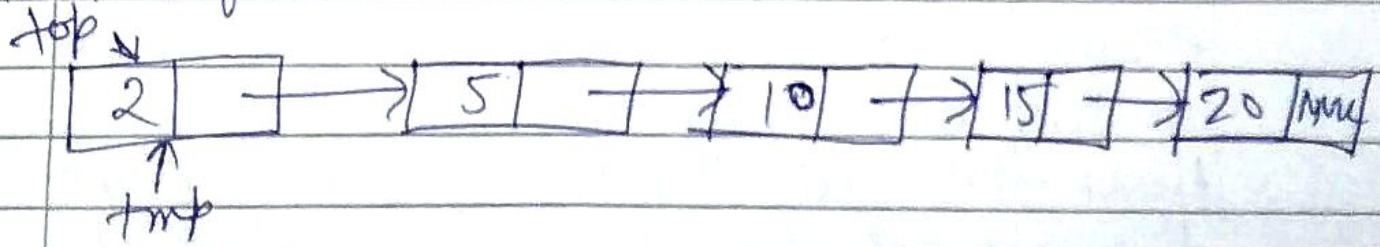


We take a pointer tmp

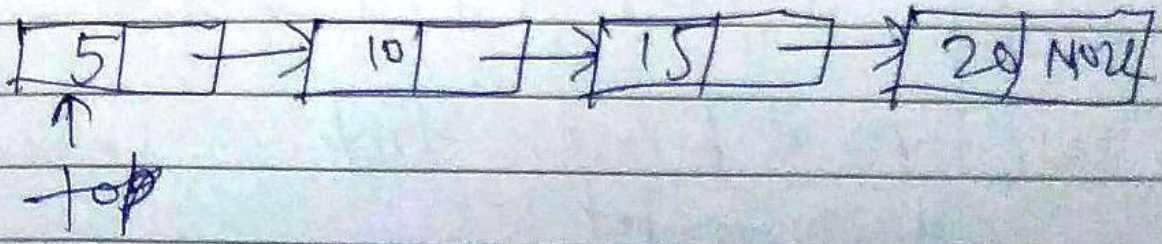
```
tmp = top
top = top -> link;
free (tmp);
```

```
struct node
{
    int info;
    struct node * link;
} *top;
void pop()
{
    struct node *tmp;
    if (top == NULL)
        printf("Stack is empty\n");
    else
    {
        tmp = top;
        printf("Popped item is %d\n",
                tmp->info);
        top = top->link;
        free(tmp);
    }
}
```

Stack before deletion.



After deletion.



```
/* Program of stack using linked list*/
# include<stdio.h>
# include<malloc.h>

struct node
{
        int info;
        struct node *link;


} *top=NULL;

main( )
{
        int choice;
```

```
        while(1)
        {
                printf("1.Push\n");
                printf("2.Pop\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ") ;
                scanf("%d", &choice);

                switch(choice)
                {
                case 1:
                        push( );
                        break;
                case 2:
                        pop( );
                        break;
                case 3:
                        display( );
                        break;
                case 4:
                        exit(1);
                default :
                        printf("Wrong choice\n");
                }/*End of switch */
        }/*End of while */
}/*End of main( ) */

push( )
{
        struct node *tmp;
        int pushed_item;
        tmp = (struct node *)malloc(sizeof(struct node));
        printf("Input the new value to be pushed on the stack : ");
        scanf("%d",&pushed_item);
        tmp->info=pushed_item;
        tmp->link=top;
        top=tmp;
}/*End of push( )*/

pop( )
{
        struct node *tmp;
        if(top == NULL)
                printf("Stack is empty\n");
        else
        {       tmp=top;
                printf("Popped item is %d\n",tmp->info);
                top=top->link;
```

```
                free(tmp);
        }

}/*End of pop( )*/

display( )
{       struct node *ptr;
        ptr=top;
        if(top==NULL)
                printf("Stack is empty\n");
        else
        {
                printf("Stack elements :\n");
                while(ptr!= NULL)
                {
                        printf("%d\n",ptr->info);
                        ptr = ptr->link;
                }/*End of while */
        }/*End of else*/
}/*End of display( )*/
```

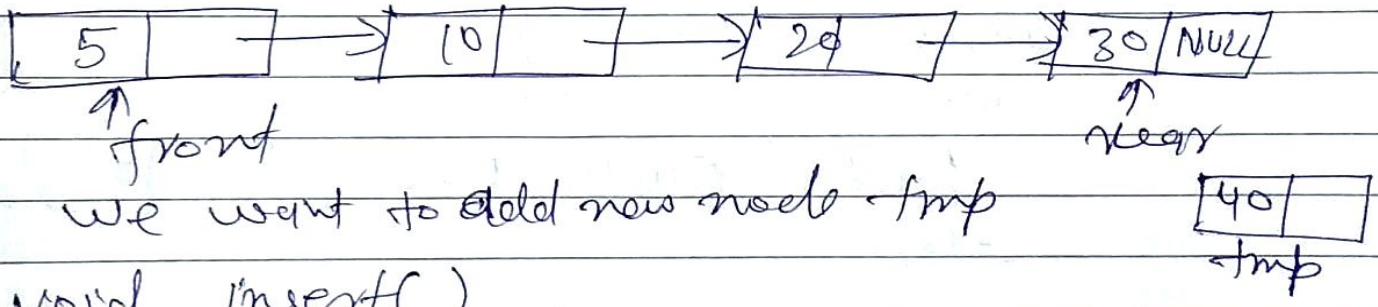
Linked List implementation of queue

Queue can also be implemented through linked list. Structure used is

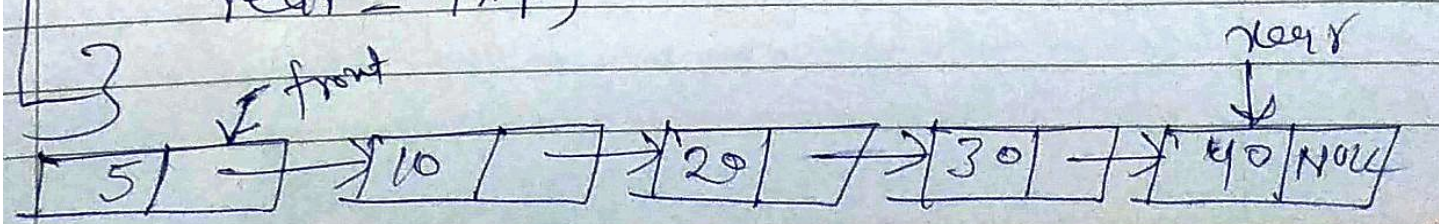
```
struct node
{
    int info;
    struct node *link;
};
```

Add operation in queue

For adding the element in queue we add the element at the end of linked list. Here front will point to the first node of linked list and rear will point to the last node of linked list.

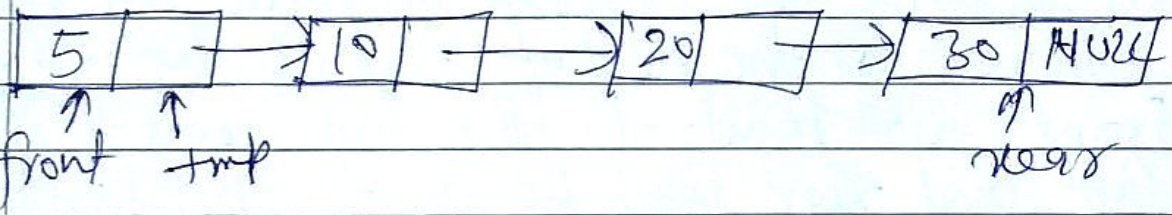



```
void insert()
{
    struct node *tmp;
    int m;
    tmp = (void *) malloc (sizeof(struct node));
    printf("\n Enter value of m for insertion \n");
    scanf("%d", &m);
    tmp -> info = m;
    tmp -> link = NULL;
    if (front == NULL)          // queue is empty.
        front = tmp;
    else
        rear -> link = tmp;
    rear = tmp;
}
```



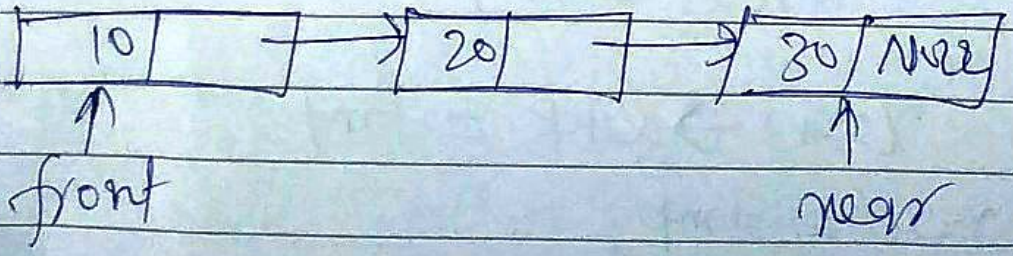
Delete operation in queue

For deleting the element of a queue, we delete the first node of the linked list. Queue is a first in first out structure and first node of linked list will be the first element of the queue.




```
void del()
{
  struct node *tmp;
  if(front==NULL)
    printf("Queue Underflow \n");
  else
  {
      tmp=front;
      printf("Deleted element is %d",tmp->info);
      front=front->link;
      free(tmp);
  }
}
```

Now the queue is



```
/* Program of queue using linked list*/
# include<stdio.h>
# include<malloc.h>

struct node
{
        int info;
        struct node *link;
}*front=NULL,*rear=NULL;

main( )
{
        int choice;
        while(1)
        {
                printf("1.Insert\n");
                printf("2.Delete\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ");
                scanf("%d", &choice);

                switch(choice)
                {
                case 1:
                        insert( );
                        break;
                case 2:
```

```
                del( );
                break;
        case 3:
                display( );
                break;
        case 4:
                exit(1);
        default :
                printf("Wrong choice\n");
        }/*End of switch*/
    }/*End of while*/
}/*End of main( )*/

insert( )
{
    struct node *tmp;
    int added_item;
    tmp = (struct node *)malloc(sizeof(struct node));
    printf("Input the element for adding in queue : ");
    scanf("%d",&added_item);
    tmp->info = added_item;
    tmp->link=NULL;
    if(front==NULL)                    /*If Queue is empty*/
        front=tmp;
    else
        rear->link=tmp;
    rear=tmp;
}/*End of insert( )*/

del( )
{
    struct node *tmp;
    if(front == NULL)
        printf("Queue Underflow\n");
    else
    {
        tmp=front;
        printf("Deleted element is  %d\n",tmp->info);
        front=front->link;
        free(tmp);
    }
}/*End of del( )*/

display( )
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
        printf("Queue is empty\n");
```

```
        else
        {
                printf("Queue elements :\n");
                while(ptr != NULL)
                {
                        printf("%d ",ptr->info);
                        ptr = ptr->link;
                }
                printf("\n");
        }/*End of else*/
}/*End of display( )*/
```