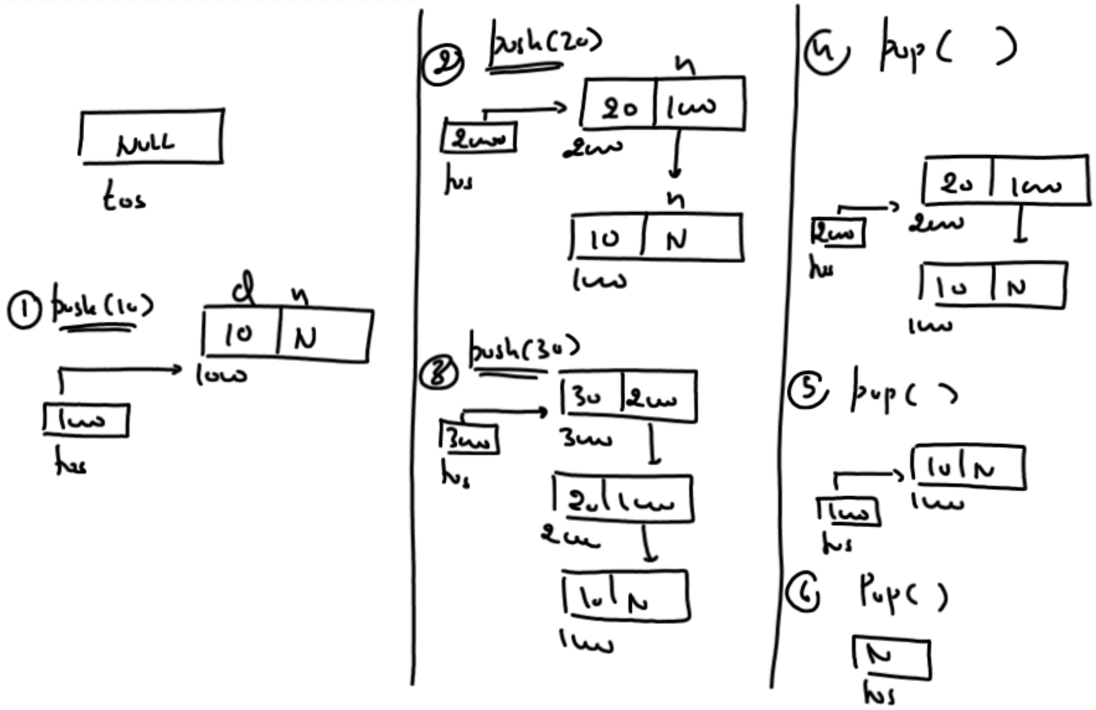
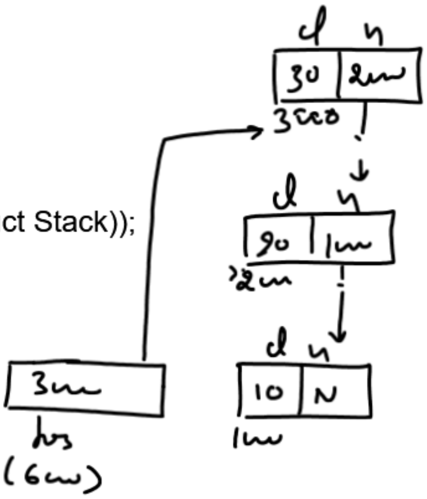
APPLICATIONS OF LINKED LIST

Implementing A Dynamic Stack
========================

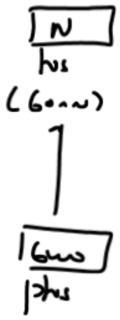


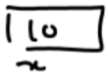
```
struct Stack
{
   int data;
   struct Stack *next;
};

void push(struct Stack **,int);
int pop(struct Stack **);
int main()
{
  struct Stack *tos=NULL;
  push(&tos,10);
  push(&tos,20);
  push(&tos,30);
  printf("\nPopped val=%d",pop(&tos);
  ....
  return 0;
}
```

```
void push(struct Stack **ptos,int x)
{
  struct Stack *p;
  p=(struct Stack *)malloc(sizeof(struct Stack));
  if(p==NULL)
   {
     printf("Stack Overflow");
     return;
   }
  p->data=x;
  p->next=*ptos;
  *ptos=p;
}
```

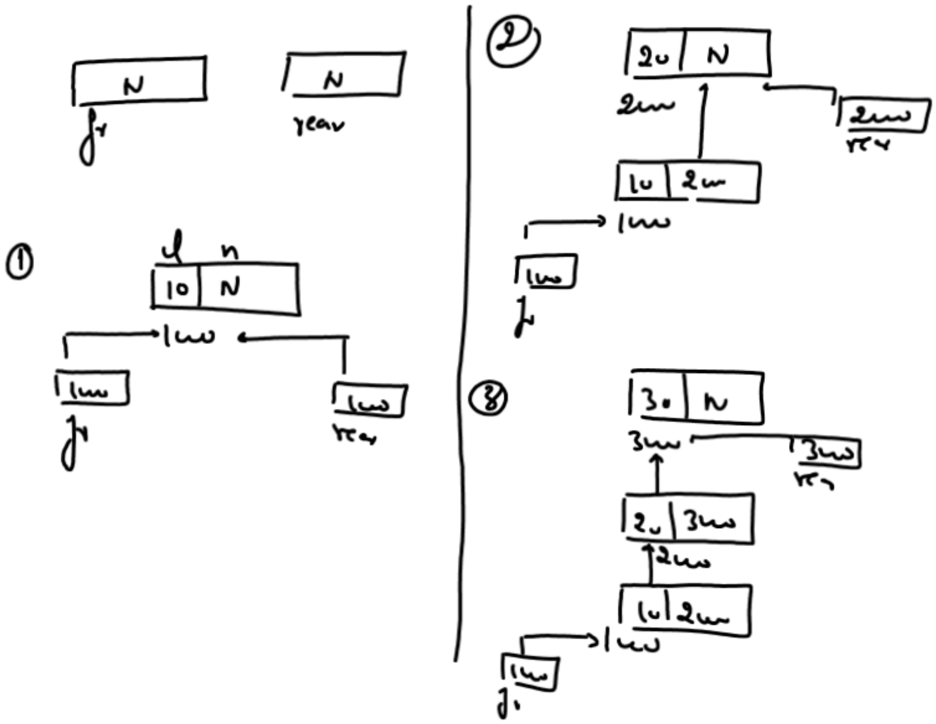


```
int pop(struct Stack **ptos)
{
   struct Stack *p;
   int x;
   if(*ptos==NULL)
    {
       printf("Stack Underflow");
       return -1;
    }
    p=*ptos;
    x=p->data;
    *ptos=p->next;
     free(p);
     return x;
}
```



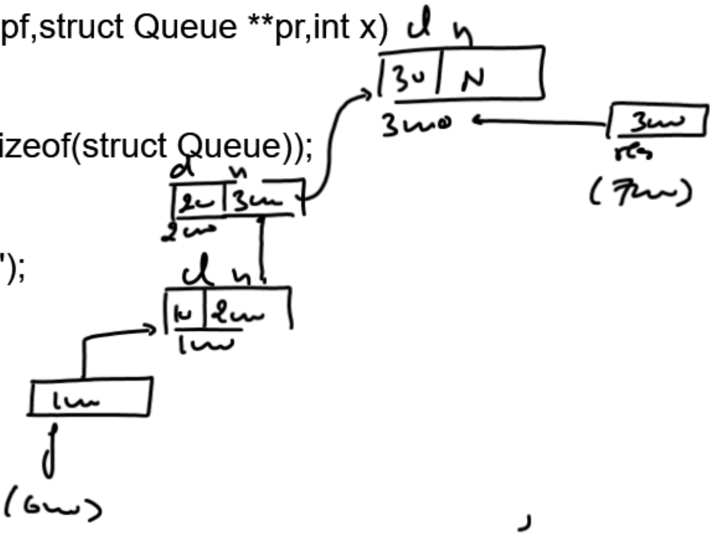


Implementing A Dynamic Queue
==========================



```
struct Queue
{
    int data;
    struct Queue *next;
};
void enqueue(struct Queue **,struct Queue **,int);
int dequeue(struct Queue **,struct Queue **);
int main()
{
   struct Queue *front,*rear;
   front=rear=NULL;
   enqueue(&front,&rear,10);
   enqueue(&front,&rear,20);
   enqueue(&front,&rear,30);
   ......
   printf("Deleted ele=%d",dequeue(&front,&rear));
   ....
   return 0;
}
```

```
void enqueue(struct Queue **pf,struct Queue **pr,int x)
{
    struct Queue *p;
    p=(struct Queue *)malloc(sizeof(struct Queue));
    if(p==NULL)
      {
        printf("Queue Overflow");
        return;
      }
    p->data=x;
    p->next=NULL;
    if(*pf)==NULL)

          *pf=p;
   else
          (*pr)->next=p;
   *pr=p;
}
```


d n
30 N
3000
3000
r
(7000)
d n
20 3000
2000
d n
10 2000
1000
1000
f
(6000)