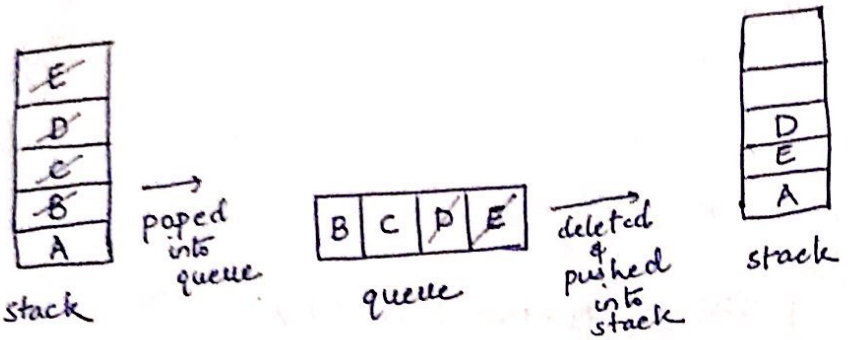
PRIYAL SHARMA
991910 2016
E1

SDF - II

TUTORIAL - 6 (QUEUE)

1.

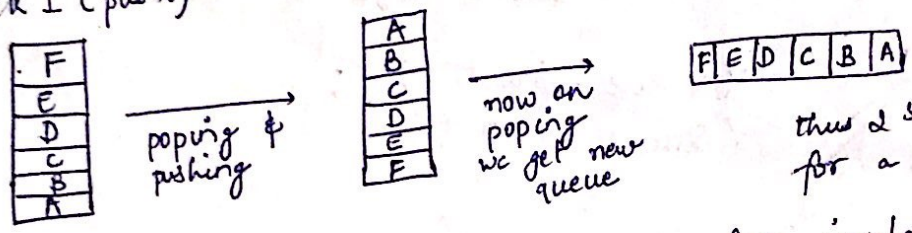


The item poped from stack is D

2. There is a requirement of 2 stacks for implementation of a queue

Eg a queue : F E D C B A

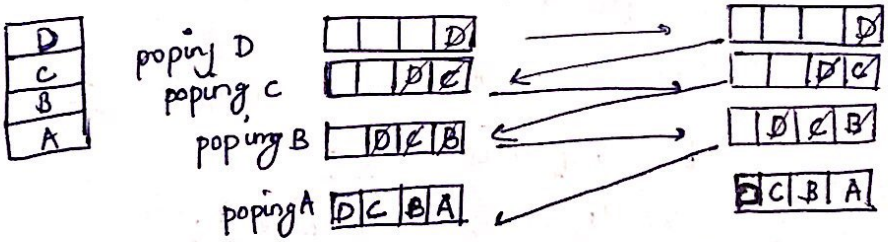
stack 1 (push)



thus 2 stacks required for a queue.

3. There is a requirement of 2 queues for implementation of a stack.

Eg a stack :



Now dequeue into stack.

we get same stack
Hence 2 queues are required

4.
```
#include <stdio.h>
#include <stdlib.h>
void reverse (int n, Queue)
```

```
int main ()
{ int n = 5;
  enqueue (10);
  enqueue (20);
  enqueue (30);
  enqueue (100
for (i=0; i<n; i++)
  { push ( queue[front]);
    dequeue ();
  }
while ( )
  { enqueue ( stack.top ) }.
  pop ();
  }
  for (i=0; i< 100-n; i++)
    { enqueue ( queue[front]);
      dequeue ();
    }
    where ( rear != 0-1)
      { printf ("%d", queue[front]);.
        queue.pop ();
      }
```

5.
```
# include <stdio.h>
int max = 50;
int queue [50];
int rear = -1, front - 1;
  void enqueue (item)
    { if (rear == max -1)
      { printf (" queue overflow");
      }
      else {
        if ( front == -1)
        { front = 0
          rear ++;
          queue [rear] = item;
        } }
        }
```

```
void dequeue ()
{ if (front == -1 || front > rear)
  printf ("ERROR");
  else { return queue (front) }
      front++;
}
int main () {
  int x
  enqueue (1);
  enqueue (2);
  enqueue (3);
  enqueue (4);
  enqueue (5);
  int s = rear;
for (i=0; i<s ; i++)
    { for (i=0; i< rear ; i++)
    { x = dequeue );
      enqueue (x);
       x = queue (front);
    }
    new enqueue (front) (1);
    dequeue ;
    }
    where ( new_ rear != -1)
    { dequeue ; }
    }
```

6

```
intervene ( int queue [9] )
{
int stack s [9]
int hs = q.size ()/2 ; i;
for (i=0; i< hs ; i++)
 { s.push ( q. front () );
   queue.pop () ;
```

```
while (! Is empty ())
{Queue q. enqueue (s. top ());
        s. pop ();
 }
for (i=0; i<hs; i++)
 {
 Queue q. enqueue (q. front ());
 Queue q. pop ();
for (i=0; i<hs; i++);
 { s. push (q. front ());
  Queue q. pop ();
while (! Is empty ())
 { Queue q. push (s. top ());
        s. pop ();
  Queue q. push (q. front ());
  Queue q. pop ();
  }
  { printf ("%d", Queue q. front ())
        Queue q. pop ();

int main ()
 { int Queue
  intervene (Q);
```

```
a[4] enqueue (4);
Queue q-enqueue (3);
        enqueue (2);
        enqueue (1);
```

7.

```
# include <stdio. h>
struct petrol pump
 { int petrol;
   int dj.
 };
int print ("struct petrol pump as[], int n)
 { int i=0; e=1, cp;
 cp = arr [r]. petrol. arr [i]. d;
 while (e! = s || cp < 0)
  { while (cp<0 & & r! =e)
  { cp = cp - arr [i]. petrol arr [s]. distance
```

```
        r = (r+1) % n;
        if (r == 0)
            return -1
    }
    cp += arr[e].petrol - arr[e].distance;
    e = (e+1) % n
    } return r;
}
int main().
{ struct petrol pump arr[]
  = {{4,8}, {6,5}, {7,3}, {4,5}};
  int n = (4/2)
  int start = pt(arr, n);
  if (start == -1)
    { printf("No solution");
  } else
      printf("start = %d", start);
  }
```