Code 1 and 2

Simple Queue : enqueue and dequeue operation

```
#include <stdio.h>
int n=1;
int k=0;
int front=-1;
int rear=-1;
int enqueue(int x,int arr[]){
  if(rear<=n){
  rear++;}
  else{
    printf("Queue overflowed\n");
    return 0;
  }
  arr[rear]=x;
  if(rear==0)
    front++;
  if(rear==n){
    printf("Queue filled\n");
  }
  0->8  1->
}
int dequeue(int arr[]){
  front++;
  if(rear<front-1){
    printf("Error");
```

```
        return 0;
    }
    else{
        return arr[front-1];
    }
}
int main()
{
    int i,arr[n+1],x,y;
    enqueue(8,arr);
    printf("1st entry done\n");
    k=k+1;
    enqueue(5,arr);
    printf("2nd entry done\n");
    k=k+1;
    enqueue(9,arr);
    printf("3rd entry done\n");
    k=k+1;
    int r;
    r=dequeue(arr);
    printf("1st deletion done%d\n",r);
    k=k-1;
    r=dequeue(arr);
    printf("2nd deletion done%d\n",r);
    k-=1;
    for(i=front;i<=rear;i++){
        printf("%d\n",arr[i]);
    }

    return 0;
}
```

Output:

```
1st entry done
Queue filled
2nd entry done
3rd entry done
1st deletion done8
2nd deletion done5
9
```

Code 3 and 4:

Cyclic Queue enqueue and dequeue operation

```
#include <stdio.h>
int n = 1;
int k = 0;
int front = 0;
int rear = -1;
int rear1;
int
enqueue (int x, int arr[])
{
  if (rear <= n)
```

```
    {
      rear++;
    }
  else
    {
      rear1 = rear;
      rear = 0;
      if (rear < front)
	{
	  rear = 0;
	}
      else
	{
	  printf ("Queue filled and overflowed");
	  rear = rear1;
	}
    }
  arr[rear] = x;
  //printf("\nrear value=%d %d\n",arr[rear],rear);
  return 0;
}

int
dequeue (int arr[])
{
  front++;
  if (rear < front - 1)
```

```
    {
      printf ("Error");
      return 0;
    }
  else
    {
      return arr[front - 1];
    }
}

int main ()
{
  int i, arr[n + 1], x, y, r;
  enqueue (8, arr);
  printf ("1st entry done\n");
  k = k + 1;
  enqueue (5, arr);
  printf ("2nd entry done\n");
  k = k + 1;
  enqueue (9, arr);
  printf ("3rd entry done\n");
  k = k + 1;
  r = dequeue (arr);
  printf ("1st deletion done %d\n", r);
  k = k - 1;

  enqueue (11, arr);
  k += 1;
   printf ("entry done\n");
  if (front < rear)
    {
      for (i = front; i <= rear; i++)
          {
            printf ("%d\n", arr[i]);
          }
    }
  else
    {
      for (i = 0; i <= rear; i++)
          {
            printf ("%d\t", arr[i]);
          }
          for(i=front;i<=n+1;i++){
              printf("%d\n", arr[i]);
          }
    }
return 0;
}
```

Output:

```
1st entry done


2nd entry done
```

```
3rd entry done

1st deletion done 8


entry done

11      5

9
```

Code 5 and 6

Double Queue operation

```
#include <stdio.h>

int n = 2;

int k = 0;

int front = -1;

int rear = -1;

int rear1;


int isfull()

{

   if((front==0)&&(rear==4)){

       return 1;

   }

   return 0;

}


int isempty()
```

```
{ if(front==-1){return 1;}

return 0;}

int enqueue(int x,int arr[]){
   if(rear<=n){
   rear++;}
   else{
      printf("Queue overflowed\n");
      return 0;
   }
   arr[rear]=x;
   if(rear==0)
      front++;
   if(rear==n){
      printf("Queue filled\n");
   }

}
int dequeue(int arr[]){
   front++;
   if(rear<front-1){
      printf("Error");
     return 0;
   }
   else{
      return arr[front-1];
```

```
    }
}



void push(int x, int arr[])
{
  if (front == -1)
    {
      front = 0;
      rear = 0;
    }
  else
    {
      front = front - 1;
    }
  arr[front] = x;
}

int eject(int arr[])
{
  int ele;
  ele = arr[rear];
  arr[rear] = 0;
  if (front == rear)
    {
      front = -1;
      rear = -1;
    }
  else
    {
      rear = rear - 1;
    }
  return ele;
}

int main ()
{
  int i, arr[n+1], x, y, r;
  push(8, arr);
  printf ("1st entry done\n");
  k = k + 1;
  printf ("%d%d\n", front, rear);

  enqueue(82,arr);
  k+1;
  printf("2nd entry done\n");
  printf ("%d%d\n", front, rear);
  enqueue (5, arr);
  printf ("3nd entry done\n");
  printf ("%d%d\n", front, rear);
  r = eject (arr);
  printf ("1st deletion done %d\n", r);
  printf ("%d%d\n", front, rear);
  k=k-1;
  r = dequeue(arr);
  printf ("second2 deletion done %d\n", r);
  arr[front-1]=0;
  k=k-1;
  printf ("%d%d\n", front, rear);

   for (i = 0; i < 3; i++)
{
 printf ("%d\n", arr[i]);
}
  return 0;
}
```

Output:

```
1st entry done
00
2nd entry done
01
Queue filled
3nd entry done
02
1st deletion done 5
01
second2 deletion done 8
11
0
82
```

0