Linear Queue

```
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>


typedef struct{

   int f,r;

   unsigned capacity;

   int *arrptr;//

}q;


q* newq(unsigned cap){

   q* new = malloc(sizeof(q));

   new->f=new->r=-1;

   new->capacity=cap;

   new->arrptr=malloc(cap*sizeof(int));

   return new;

}


bool is_full_q(q* sub){

   return (sub->r)==(sub->capacity-1);

}


bool is_empty_q(q* sub){

   return (sub->r==-1)&&(sub->f==-1);
```

```
}

bool is_single_element(q* sub){
   return((sub->r==sub->f));
}

void enq(q* sub ,int val){
   if(is_full_q(sub)) return;
   if(is_empty_q(sub)) sub->f=sub->r=0;
   else sub->r=(sub->r+1);
   sub->arrptr[sub->r]=val;
}

int deq(q* sub){
   if(is_empty_q(sub)) return -1;
   int temp=sub->arrptr[sub->f];
   if(is_single_element(sub)) sub->f=sub->r=-1;
   else sub->f=(sub->f+1);
   return temp;
}

void printq(q*sub){
   if(is_empty_q(sub)) printf(" \n");
   else{
      int i=sub->f;
```

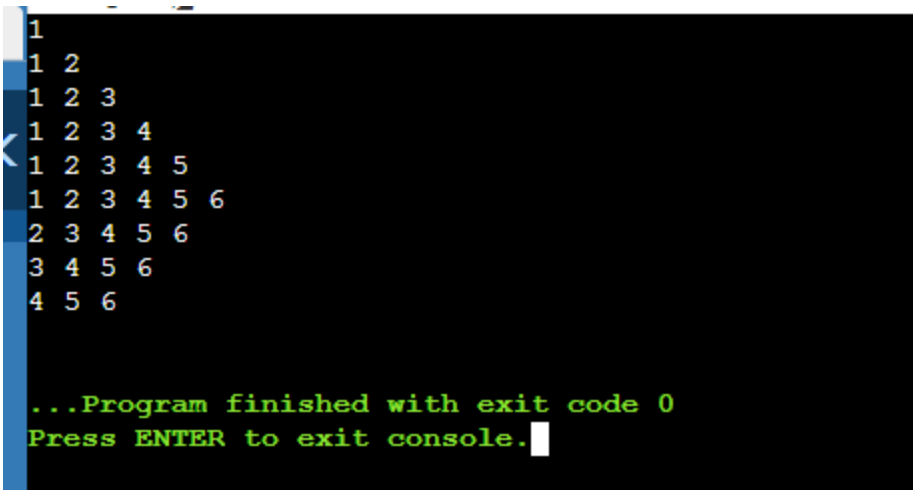
```
        do{
            printf("%d ",sub->arrptr[i]);
            i=(i+1)%sub->capacity;
        }while(i!=sub->r+1);
        printf(" \n");
    }
}



int main(){
    q* myq=newq(15);
    enq(myq,1);
     printq(myq);
    enq(myq,2);
    printq(myq);
    enq(myq,3);
    printq(myq);
        enq(myq,4);
    printq(myq);
        enq(myq,5);
    printq(myq);
            enq(myq,6);
    printq(myq);
    deq(myq);
    printq(myq);
```

```
    deq(myq);
  printq(myq);
    deq(myq);
  printq(myq);

}
```



Circular queue

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

typedef struct{
  int f,r;
  unsigned capacity;
  int *arrptr;
```

```
}q;

q* newq(unsigned cap){
    q* new = malloc(sizeof(q));
    new->f=new->r=-1;
    new->capacity=cap;
    new->arrptr=malloc(cap*sizeof(int));
    return new;
}

bool is_full_q(q* sub){
    return(((sub->r+1)%sub->capacity)==sub->f);
}

bool is_empty_q(q* sub){
    return((sub->r==-1&&sub->f==-1));
}

bool is_single_element(q* sub){
    return((sub->r==sub->f));
}

void enq(q* sub ,int val){
    if(is_full_q(sub)) return;
    if(is_empty_q(sub)) sub->f=sub->r=0;
```

```
    else sub->r=(sub->r+1)%sub->capacity;
    sub->arrptr[sub->r]=val;
}

int deq(q* sub){
    if(is_empty_q(sub)) return -1;
    int temp=sub->arrptr[sub->f];
    if(is_single_element(sub)) sub->f=sub->r=-1;
    else sub->f=(sub->f+1)%sub->capacity;
    return temp;
}

void printq(q*sub){
    if(is_empty_q(sub)) printf(" \n");
    else{
        int i=sub->f;
        do{
            printf("%d ",sub->arrptr[i]);
            i=(i+1)%sub->capacity;
        }while(i!=sub->r+1);
        printf(" \n");
    }
}
```

```
int main(){

   q* myq=newq(15);

   enq(myq,1);

   enq(myq,2);

   printq(myq);

   enq(myq,3);

   printq(myq);

      enq(myq,4);

   printq(myq);

      enq(myq,5);

   printq(myq);

   deq(myq);

   printq(myq);

      deq(myq);

   printq(myq);

      deq(myq);

   printq(myq);


}
```

```
1 2
1 2 3
1 2 3 4
1 2 3 4 5
2 3 4 5
3 4 5
4 5


...Program finished with exit code 0
Press ENTER to exit console.
```