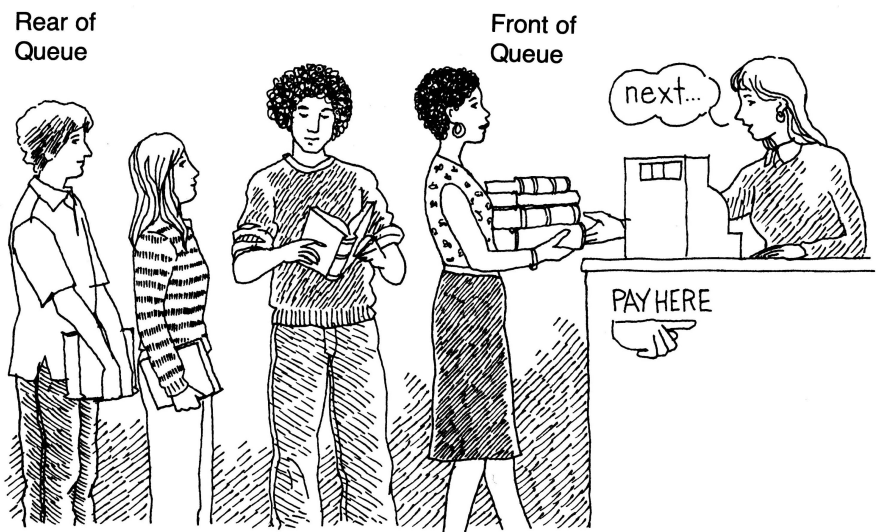
Queues

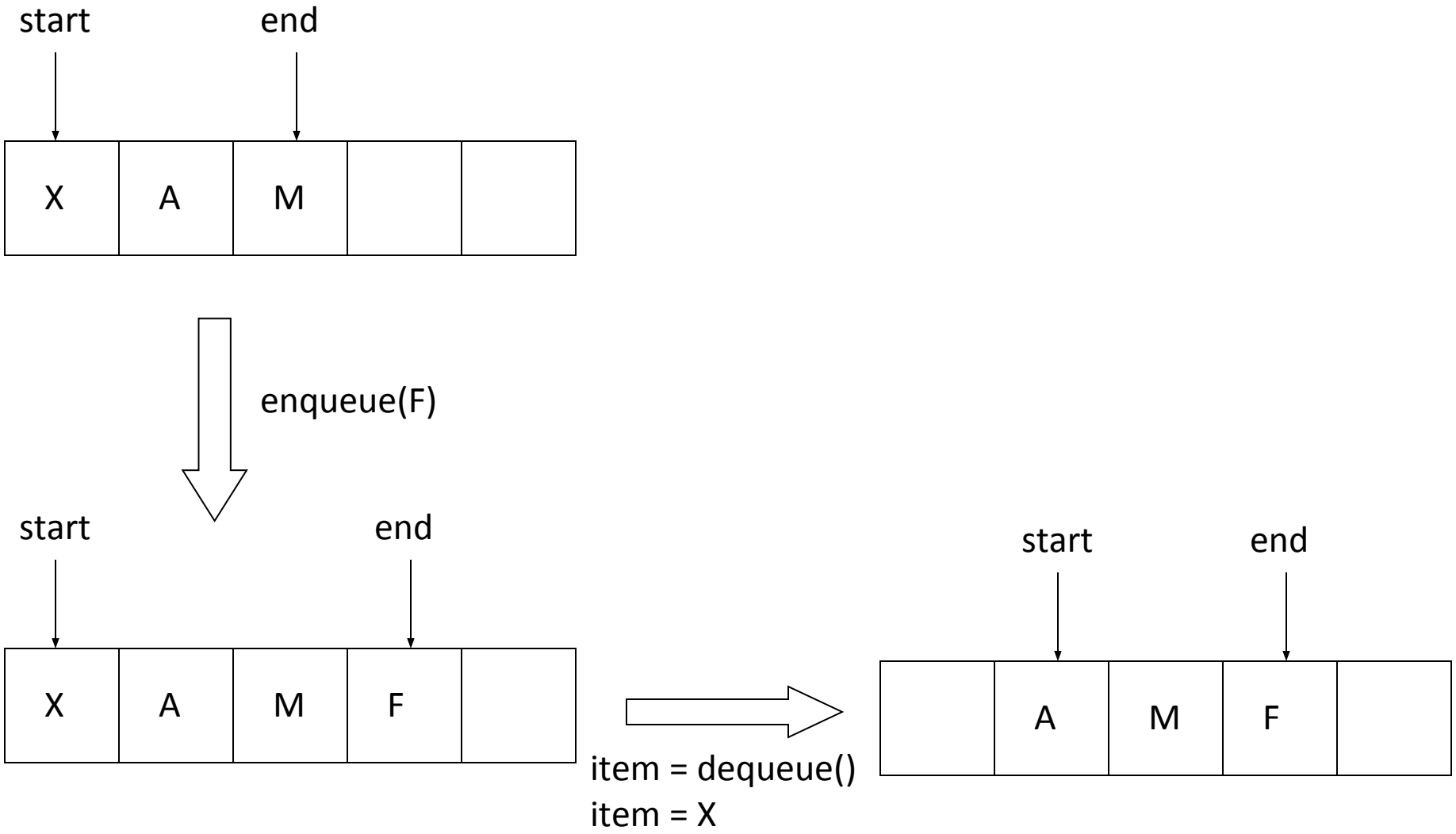
CS 308 – Data Structures

What is a queue?

- It is an ordered group of homogeneous items of elements.
- Queues have two ends:
  - Elements are added at one end.
  - Elements are removed from the other end.
- The element added first is also removed first (**FIFO**: First In, First Out).



Queue


start
end
X
A
M
enqueue(F)
start
end
X
A
M
F
item = dequeue()
item = X
start
end
A
M
F

Implementing a Queue

- At least two common methods for implementing a queue
  - array
  - linked list
- Which method to use depends on the application

Queue Specification

- <u>Definitions</u>: (provided by the user)
  - *MAX_ITEMS*: Max number of items that might be on the queue
  - *ItemType*: Data type of the items on the queue

Enqueue (ItemType newItem)

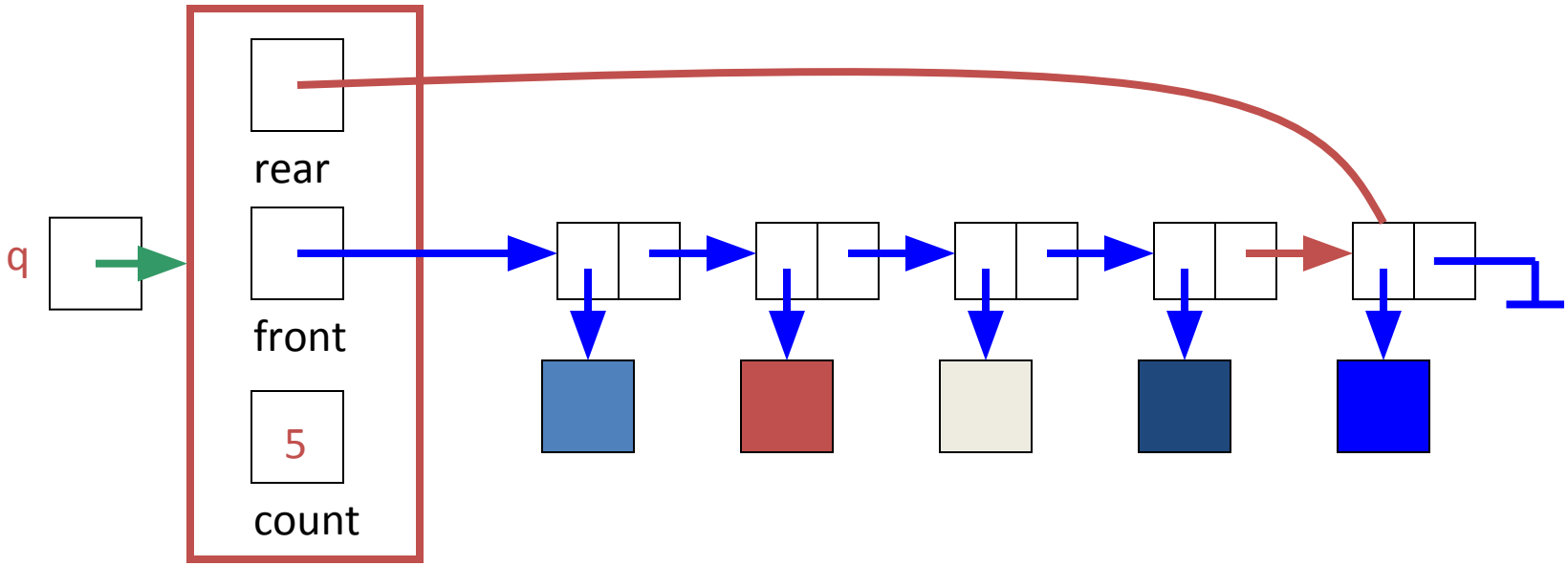
- *Function*: Adds newItem to the rear of the queue.
- *Preconditions*: Queue has been initialized and is not full.
- *Postconditions*: newItem is at rear of queue.

Dequeue (ItemType& item)

- *Function*: Removes front item from queue and returns it in item.
- *Preconditions*: Queue has been initialized and is not empty.
- *Postconditions*: Front element has been removed from queue and item is a copy of removed element.

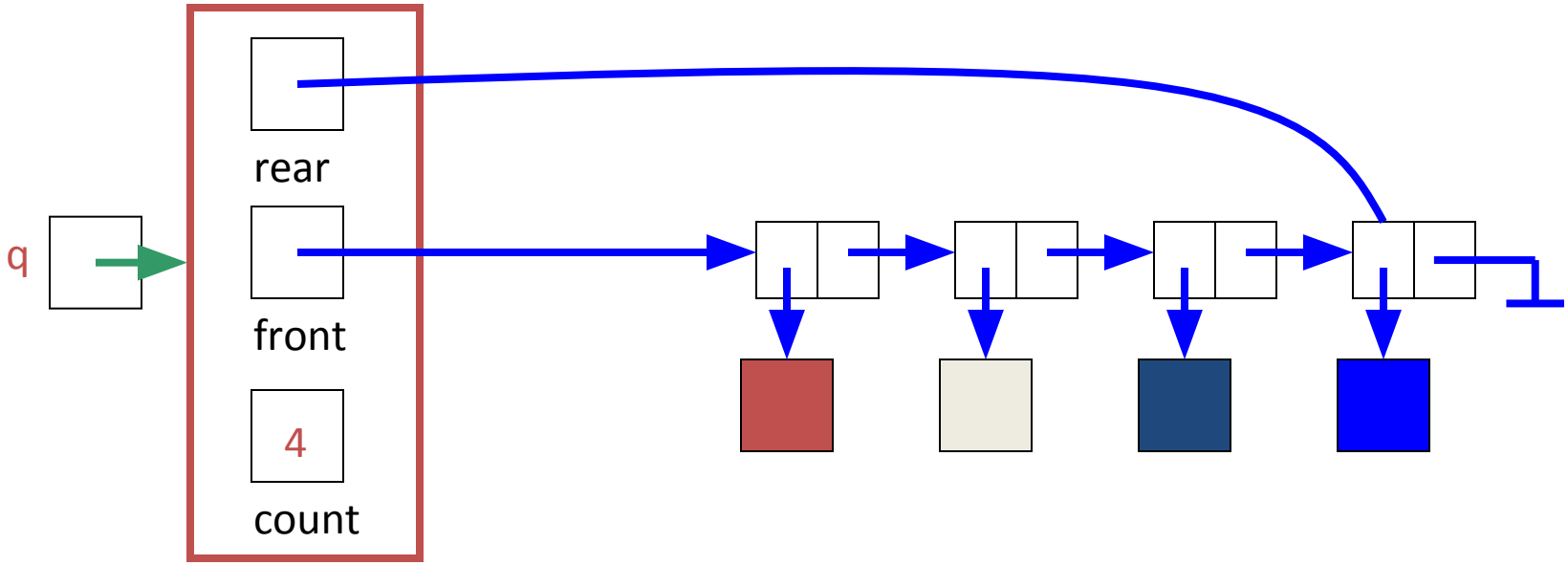
Queue After Adding Element

New element is added in a node at the end of the list, rear points to the new node, and count is incremented



Queue After a **dequeue** Operation

Node containing i[illegible]oved from the front of the list (see previous slide), front now points to the node that was formerly second, and count has been decremented.



Insert an item at the rear end of queue

```
void insert_rear()
{
 if ( rear== size-1)
    {
        printf("Queue is overflow\n");
        return;
    }
 rear =rear+1;
q[rear]=item;
}
```

Delete an item from the front end

- int delete_front()

- if(front>rear)
- {
- return -1;
- return q[front++];
- }

Display the queue elements

```
void display()
{
int i;
if(front>rear)
{
printf("Queue is empty");
return;
}
printf("Contents of the queue are");

for(i=front;i<=rear;i++)
{
printf("%d\n",q[i];
}
```

6.A C program to simulate the working of Messaging System in which a message is placed in a Queue by a Message Sender, a message is removed from the queue by a Message Receiver, which can also display the contents of the Queue.

```
#include <stdio.h>
#include<string.h>
#define size 5
int f=0,r=-1;

struct Queue
{
   int id;
   char msg[20];
}q[size];
```

```
void insert()
{
    int x;
    char txt[20];
    if ( r == size-1)
    {
        printf("Queue is full\n");
        return;
    }
     //if (f == -1) f =0;
     printf(" entser the id and messege\n");
     scanf("%d", &x);
     gets(txt);
     r++;
     q[r].id = x;
     strcpy(q[r].msg, txt);
     printf("%d msg is inserted\n", x);
}
```

```
void delete()
{
   if(f>r )
   {
      printf(" Queue is empty\n");
      f=0; r=size-1;
      return;
   }
    printf(" %d is deleted\n", q[f++].id);
   }

void display()
{
   int i;
    if(f>r)
   {
      printf(" Queue is empty\n");

      return;
   }
   for(i= f; i<=r;i++)
   printf(" %d\t %s \n", q[i].id,q[i].msg);
   printf("\n");
}
```

```
int main()
{
   int ch;
   printf("1: Insert\n2:Delete\n3: Display\n4:Exit\n");
   while(1)
   {
      printf("Enter yout choice\t");
      scanf("%d", &ch);
      switch(ch)
      {
         case 1: insert();
            break;
         case 2: delete();
            break;
         case 3: display();
            break;
         case 4: exit(0);

         default : printf("Invalid choice\n");

      }
   }
```