DATA STRUCTURES AND ALGORITHMS

Lecture 7: Queues

Lecturer: Mohsin Abbas

National University of Modern Languages, Islamabad

QUEUES

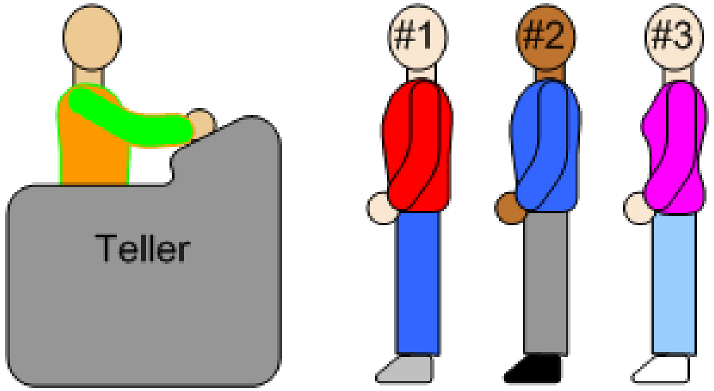
- Queue is ***First-In-First-Out (FIFO)*** data structure
  - *First element added* to the queue will be *first one to be removed*.

- Queue implements a special kind of list.
  - Items are inserted at one end (the rear).
  - Items are deleted at the other end (the front)

QUEUES

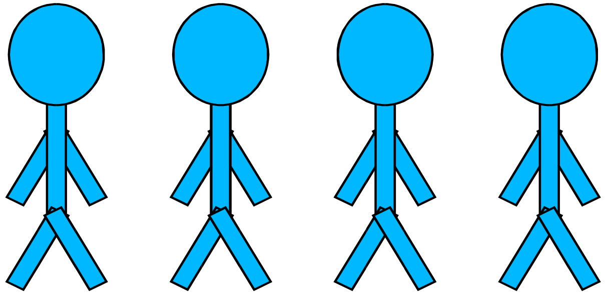
Examples:

- A queue is like a line of people waiting for a bank teller.
- The queue at billing counter.



QUEUES

- The queue has a front and a rear.

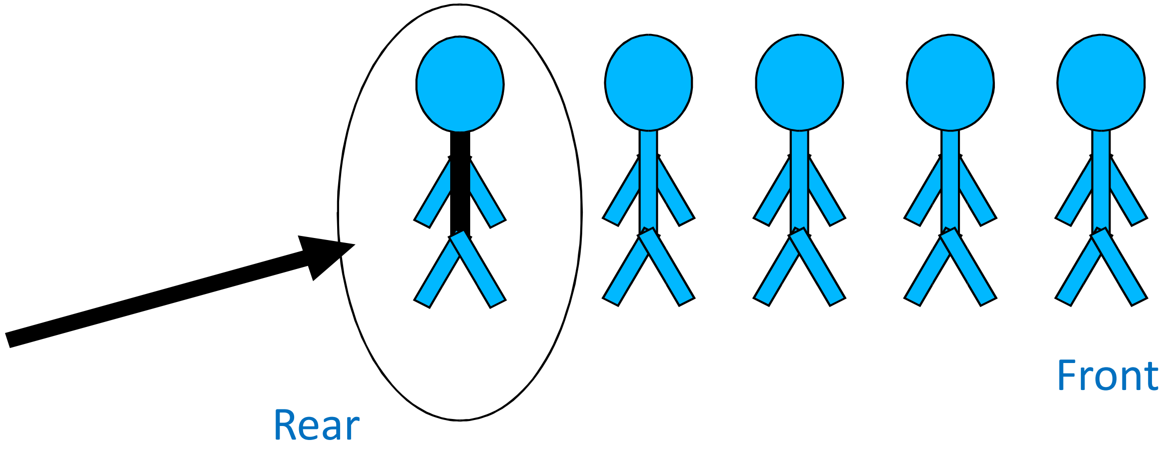


Rear

Front

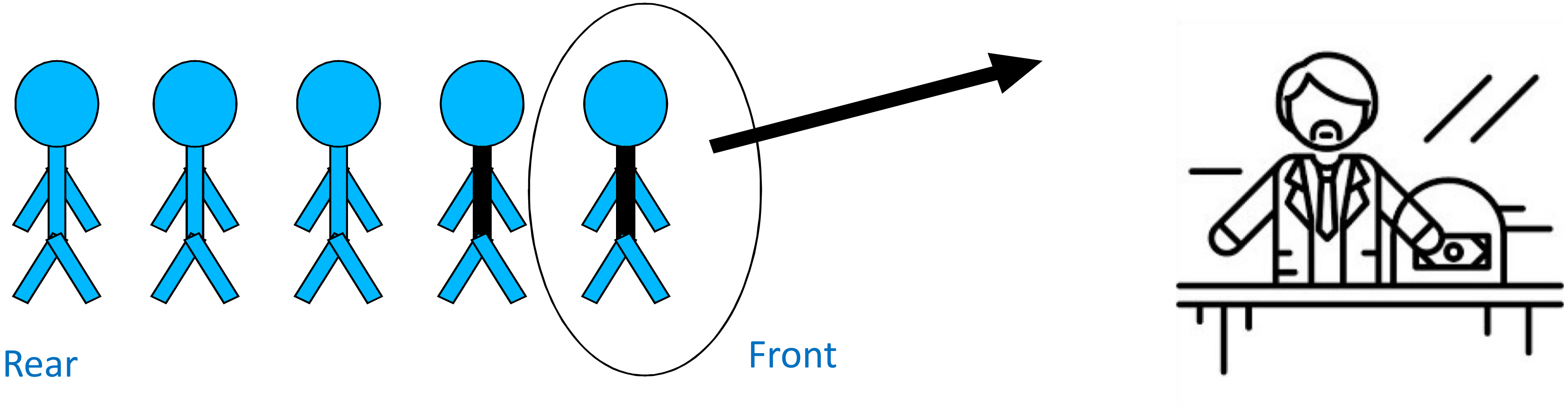
QUEUES

- New people must enter the queue at the rear.



QUEUES

- An item is always leave from the front of the queue.



QUEUES – EXAMPLES

- Billing counter.
  - Booking movie tickets.
  - Queue for paying bills.
- A print queue.
- Vehicles on toll-tax bridge.
- Luggage checking machine.
- Other examples????

QUEUES APPLICATIONS

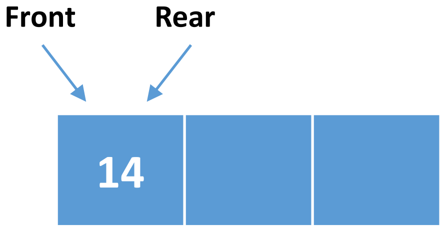
- Operating systems
  - *Process scheduling* in multiprogramming environment
  - Controlling *provisioning of resources* to multiple users (or processing)
- Middleware/Communication software
  - Hold messages/packets in order of their arrival
    - Messages are usually transmitted faster than the time to process them
  - The most common application is in client-server models
    - Multiple clients may be requesting services from one or more servers
    - Some clients may have to wait while the servers are busy
    - Those clients are placed in a queue and serviced in the order of arrival

BASIC OPERATIONS OF QUEUE

- MAKENULL(Q)
  - Make Queue Q be an empty list
- FRONT(Q)
  - Return the first element of Queue Q
- ENQUEUE(x, Q)
  - Insert the element x at the end of Queue Q
- DEQUEUE(Q)
  - Remove the first element of the Queue Q
- EMPTY(Q)
  - Return true if and only if Q is an empty Queue and return false otherwise

ENQUEUE AND DEQUEUE OPERATIONS

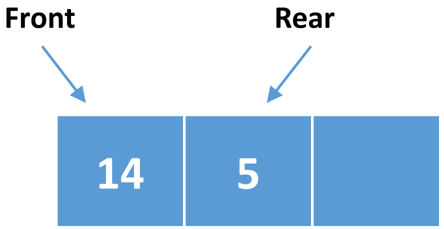
ENQUEUE(14);



Front Rear

14

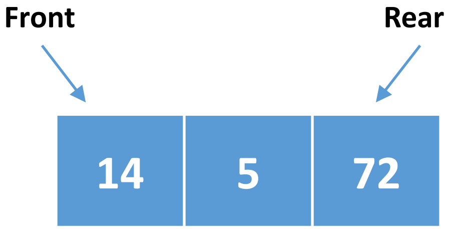
ENQUEUE(5);



Front Rear

14 5

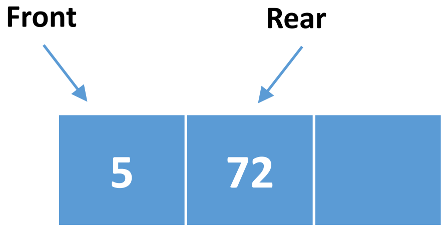
ENQUEUE(72);



Front Rear

14 5 72

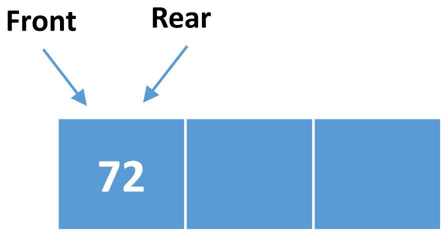
DEQUEUE();



Front Rear

5 72

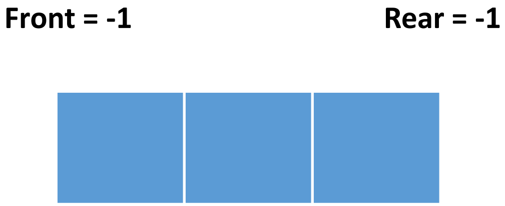
DEQUEUE();



Front Rear

72

DEQUEUE();



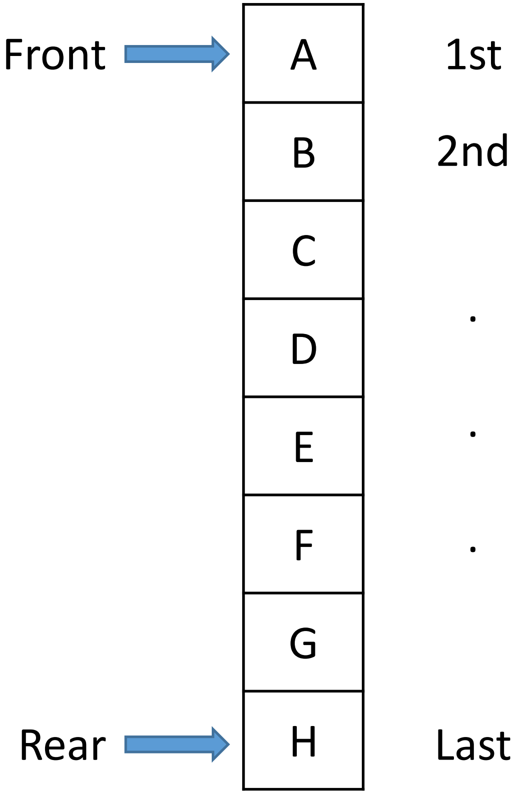
Front = -1 Rear = -1

QUEUE IMPLEMENTATION

- Implementation of queue can be done in two ways
  - Static implementation
  - Dynamic Implementation
- Static Implementation
  - Queue is implemented by *arrays*
  - Size of queue remains fix
- Dynamic Implementation
  - A queue can be implemented as a *linked list*
  - Expand or shrink with each enqueue or dequeue operation

STATIC IMPLEMENTATION

- Use *two counters* that signify rear and front.
- When queue is *empty*
  - Both *front* and *rear* are set to -1
- When there is *only one value* in the Queue,
  - Both *rear* and *front* have same index
- While *enqueueing,* increment rear by 1
- While *dequeueing*, increment front by 1



STATIC IMPLEMENTATION USING ARRAY

0	1	2	3	4	5	6	7	8

Front = -1

Rear = -1

5								
0	1	2	3	4	5	6	7	8

Front = 0

Rear = 0

5	**1**							
0	1	2	3	4	5	6	7	8

Front = 0

Rear = 1

STATIC IMPLEMENTATION USING ARRAY

5	**1**	**72**	**14**	**12**	**26**	**59**		
0	1	2	3	4	5	6	7	8

Front = 0
Rear = 6

				12	**26**	**59**		
0	1	2	3	4	5	6	7	8

Front = 4
Rear = 6

					26	**59**	**110**	**46**
0	1	2	3	4	5	6	7	8

Front = 5
Rear = 8

Problem: How can we insert more elements?
Because rear index cannot go beyond the last element…

USING CIRCULAR QUEUE

- Allow rear to wrap around the array

```
if (rear == queueSize - 1)
   rear = 0;
else
   rear++;
```

- Alternatively, use modular arithmetic

```
rear = (rear + 1) % queueSize;
```

STATIC IMPLEMENTATION USING ARRAY

					26	**59**	**110**	**46**
0	1	2	3	4	5	6	7	8

Front = 5
Rear = 8

```
ENQUEUE(86);
Rear = (Rear + 1) mod queueSize = (8 + 1) mod 9 = 0
```

86					**26**	**59**	**110**	**46**
0	1	2	3	4	5	6	7	8

Front = 5
Rear = 0

Problem: How to avoid overwriting an existing element?

HOW TO DETERMINE EMPTY AND FULL QUEUES?

- A counter indicating number of values/items in the queue
  - Covered in first array-based implementation *(Simple)*
- Without using an additional counter *(only relying on front and rear)*
  - Covered in alternative array-based implementation *(Circular)*

IMPLEMENTATION CODE

```
#include<iostream>
using namespace std;
#define SIZE 20
int a[SIZE];
int front=0;
int rear=0;

void display()
{
    cout<<"\n";
    for (int i=front;i<rear;i++)
        cout<<"\t"<<a[i];
}
void enqueue(int i)
{
    if(rear >= SIZE)
        cout<<"\nQUEUE IS FULL\n";
    else
        a[rear++] = i;
}
```

```
void dequeue()
{
    if(front == rear)
        cout<<"\nQUEUE IS EMPTY\n";
    else
    {
        for (int i=0;i<rear;i++)
            a[i] = a[i+1];
        rear--;
    }
}
int main()
{
    int option;
    char choice;
    cout<<"Implementation of Queue using
Array, Maximum Size of Queue is
"<<SIZE<<endl<<endl;
    cout<<"Choose any of the following option
"<<endl<<endl;
```

IMPLEMENTATION CODE

```
do
 {
  cout<<"\n1. INSERTION";
  cout<<"\n2. DELETION";
  cout<<"\n3. EXIT";
  cout<<"\n\nENTER YOUR CHOICE: ";
  cin>>option;
  switch (option)
  {
    case 1:
      int n;
      cout<<"How many Elements you want to
insert: ";
      cin>>n;
      cout<<"\nENTER "<<n<<" elements in
queue:\n";
      for (int x=0;x<n;x++)
      {
        int num;
        cin>>num;
        enqueue(num);
      }
      cout<<"Elements inserted in the queue
are "<<endl;
      display();
      break;
    case 2:
      int n1;
      cout<<"\nHow many elements you want to
remove from the queue? :";
      cin>>n1;
      for (int y=0;y<n1;y++)
        dequeue();
      cout<<"\nQueue after removal of
"<<n1<<" elements"<<endl;
      display();
      break;
```

IMPLEMENTATION CODE

```
    case 3:
      exit(0);
    default:
      cout<<"Invalid Choice...";
  }
 cout<<"\n\nDo you want to repeat the
program? Enter Y/N: ";
 cin>>choice;
} while (choice == 'y' || choice == 'Y');
return 0;
}

OUTPUT:
Implementation of Queue using Array, Maximum
Size of Queue is 20
Choose any of the following option
1. INSERTION
2. DELETION
3. EXIT
```

```
ENTER YOUR CHOICE : 1
How many Elements you want to insert: 4

ENTER 8 elements in queue:
5
14
110
12
72
59
46
39

Elements inserted in the queue are

5   14  110  12  72  59  46  39

Do you Repeat? Enter y/n or Y/N: n
Program END
```

CONCLUSION

- In this lecture we have studied:
  - Queue Data Structure
  - Operations of Queue
  - Static Implementation of Queue

Question?