Priority Queue ADT
Priority Queue

An arrangement of data elements that allows the insertion of data elements in any order but allows to remove data elements according to their priority values

- The highest priority element is always at the front of the queue and is removed first of all
- The highest priority can be either the minimum value of all the items, or the maximum value
- We will assume the highest priority is the maximum value
- Note that in a priority queue "first in first out" does not apply in general.
Why Priority Queue ADT?

There are many practical situations which employ priority queue

- Scheduler for an Operating System
- Printer Queue
- Sorting
  - Heapsort
- Shortest path Graph algorithms (Greedy Algorithms)
The concept of Priority

Suppose that you have a few assignments from different courses. Which assignment will you want to work on first?

<table>
<thead>
<tr>
<th>Course</th>
<th>Priority</th>
<th>Due day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Systems</td>
<td>2</td>
<td>October 3</td>
</tr>
<tr>
<td>UNIX</td>
<td>4</td>
<td>October 10</td>
</tr>
<tr>
<td>Data Structure &amp; Algorithm</td>
<td>1</td>
<td>September 29</td>
</tr>
<tr>
<td>Structured Systems Analys is</td>
<td>3</td>
<td>October 7</td>
</tr>
</tbody>
</table>

You set your priority based on due days.

When you want to determine the priority for your assignments, you need a value for each assignment that you can compare with each other.
**Priority Queue ADT**

**Elements:** Any data type

**Structure:** Any arrangement that allows to insert data elements in any order but removal of the data element with the highest priority

**Domain:** Number of elements is bounded

**Operations:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Specification</th>
</tr>
</thead>
</table>
| bool full()     | **Precondition/Requires:** none  
**Processing/Results:** returns true if the priority queue is full otherwise false. |
| int length()    | **Precondition/Requires:** none  
**Processing/Results:** returns the number of elements currently in the priority queue. |
| void enqueue(Typ, int) | **Precondition/Requires:** priority queue is not full.  
**Processing/Results:** inserts a given element into the queue according to its priority. |
| Typ serve(int &p) or Typ deque(int &p) | **Precondition/Requires:** priority queue is not empty  
**Processing/Results:** removes the element at the front or head of the priority queue and returns it and its priority as p. |
representation of priority queue ADT

Queue ADT can be represented as
- Array
- Linked List
- Heap
Linked List based Implementation of Priority Queue

```java
public class Priority
{
    int value;
    public Priority(int p)
    {
        value = p;
    }
    public int get_value()
    {
        return value;
    }
    public void set_value(int v)
    {
        value = v;
    }
}

public class PQNode<T> 
{
    private T data;
    private Priority priority;
    public PQNode<T> next;
    public PQNode()
    {
        next = null;
    }
    public PQNode(T e, Priority p) {
        data = e;
        priority = p;
    }
    public Priority get_priority() {
        return priority;
    }
    public T get_data()
    {
        return data;
    }
    public void set_priority(Priority p){
        priority = p;
    }
    public void set_data(T e){
        data = e;
    }
}
```
Linked List based Implementation of Priority Queue

```java
public class LinkPQ<T> {
    // Data Members
    private int size;
    private PQNode<T> head;
    // Operations
    public LinkPQ()
    public int length ()
    public boolean full ()
    public void enqueue(T e, Priority pty)
    public T serve (Priority pty)
}
```
Implementation of Operations

```java
public LinkPQ()
{
    head = tail = null;
    size = 0;
}

public int length ()
{
    return size;
}

public boolean full ()
{
    return false;
}
```
public void enqueue(T e, Priority pty) {
    PQNode<T> p, q, tmp;
    if ((size == 0) || (pty.get_value() >
        head.get_priority().get_value())) {
        tmp = new PQNode<T>(e, pty);
        tmp.next = head;
        head = tmp;
    }
    else {
        p = head; q = null;
        while ((p != null) &&
            (p.get_priority().get_value() > pty.get_value())) {
            q = p; p = p.next;
        }
        tmp = new PQNode<T>(e, pty);
        tmp.next = p;
        q.next = tmp;
    }
    size++;
}
public T serve (Priority pty) {
   if(length() == 0) return null;
   T e = head.get_data();
   pty.set_value(head.get_priority().get_value());
   head = head.next;
   size--;
   return e;
}
Performance Analysis

- Time complexity of all functions is $O(1)$. The time complexity of enqueue is $O(n)$.
- Since the time complexity of enqueue operation is $O(n)$, so the time required to build a priority queue with $n$ elements is $O(n^2)$.
- In Queue ADT, the enqueue operation is $O(1)$, but in priority queue this is $O(n)$, why? enqueue now requires an insertion in the queue based on priority.
Exercise
Implement priority queue using array and discuss the complexity.