# COMSC-165 Lecture Topic 13
# Ordered Sets as Linked Lists

### Reference
Wiki

### Ordered Sets
array or linked list where elements/nodes
  are arranged in order
"sorting" means to arrange in order

### Considerations In Sorting
order array elements or linked list nodes *lo-to-hi*
  but can be hi-to-lo
comparing elements and nodes
  what to compare?
  `strcmp` function
    return <0, 0, or >0 -- *NOT -1 and +1*
  dealing with "ties"
options
  *build* in order
  *rearrange* in order

### Basic Sorting Algorithms
insertion sort
  build in order
  rearrange in order
selection sort

### Building in Order
a 4-step process

```
  // STEP 1: create a new node
  tod* t = new tod;

  // STEP 2: put data into the new node
  t->hour = ...
  ...

  // STEP 3: find insertion point in new list
  tod* p, *prev;
  for (p = start, prev = 0; p; prev = p, p = p->next)
    if (todCmp(t, p) < 0) // if "t" comes before "p"...
      break; // "t" goes between "prev" and "p"

  // STEP 4: insert "t" between "prev" and "p"
  t->next = p; // "t" inserted before "p"
  if (prev) // "t" inserted after "prev"
    prev->next = t;
  else // "t" added to front of list
    start = t;
}
```

### Sorting A Linked List
build in order -- insertion sort
  get a new node to insert
  traverse to find insertion point
  insert at insertion point
rearrange in order ("insertion sort", too)
  get a node from old list to insert into new
  traverse to find insertion point
  insert at insertion point

### Rearranging In Order
rearrange in order -- insertion sort

```
tod* newStart = 0; // a new empty list
while (start)
{
  // STEPS 1+2: take a node from the old, unordered list
  tod* t = start; // get node from old list
  start = start->next; // remove start node from old list

  // STEP 3: find insertion point in new list
  tod* p, *prev;
  for (p = newStart, prev = 0; p; prev = p, p = p->next)
    if (todCmp(t, p) < 0) // if "t" comes before "p"...
      break; // "t" goes between "prev" and "p"

  // STEP 4: insert "t" between "prev" and "p"
  t->next = p; // "t" inserted before "p"
  if (prev) // "t" inserted after "prev"
    prev->next = t;
  else // "t" added to front of list
    newStart = t;
}
start = newStart; // hand off newly reordered list
```

build in order: slight variation of above
  without a newStart
  with new node creation

### A Nested For-Loop Sort Code Block
selection sort, using "swap" from the "algorithm" library

```
  for (tod* p = start; p; p = p->next)
    for (tod* q = p->next; q; q = q->next)
      if (todCmp(q, p) < 0) // then swap node contents
      {
        swap(*p, *q); // swap the contents AND pointers
        swap(p->next, q->next); // swap back the pointers
      }
```