Topics since last test

- More invariants
- Pixmap
- Sorting & Templates
  - Selection sort
  - Insertion sort
  - Quicksort
  - Loop invariants
  - Complexity
  - Function templates
  - Operator overloading
- Dynamic data
  - Pointers
  - The new and selector (->) operators
  - Passing as parameters
  - Sharing and copying
- Inheritance
  - Extending another class
  - "Is a" relationship
  - Polymorphism
  - Employee example

The exam

- Saturday, May 4, 9am-12 noon in White Lecture Hall
- Almost half multiple choice
- Cumulative
- By Monday, April 28 at 9am:
  - All grades up
  - All solutions out
  - Grade problems: submit_cps006 problems issues.txt
- Final grades up Sunday May 6 at noon

What is Computer Science?

What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple --- it is the art of programming a computer. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

C.A.R (Tony) Hoare

Computer Science vs. Engineering

- Science
  - Describe
  - Explain
- Engineering
  - Build
    "An engineer can do for a dime what any fool can do for a dollar"
- Today, CS is mostly engineering
Essential concepts

There is beauty at all levels of sophistication and all levels of abstraction.
- David A. Blackwell

If life were really fair, algebra would actually come in handy
- Amstel Light commercial

On programming and deadlines

Observe that for the programmer, as the chef, the urgency of the patron may govern the scheduled completion of task, but it cannot govern the actual completion. An omelette, promised in two minutes, may appear to be progressing nicely. But when it has not set in two minutes, the customer has two choices -- wait or eat it raw. Software customers have the same choices.
- Fred Brooks

We don’t have time to stop for gas -- we’re already late.
- Old software project planning proverb via Mike Cleron

I love deadlines. I like the whooshing sound they make as they fly by.
- Douglas Adams

Why is programming fun?

What delights may its practitioner expect as a reward?
First is the sheer joy of making things

Second is the pleasure of making things that are useful
Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts

Fourth is the joy of always learning
Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff.

Fred Brooks

On education

The college you attend does not determine the scope and possibility of your life’s achievements. It will have some influence, no doubt. What is more important is the encouragement that we, as parents and friends, offer these prospective students as they explore their own educational trail. In the end, the experiences they encounter and the depth of character they build along the way will mean far more than the name of the institution on their diploma.
- John Hennesy

Education is not filling a bucket but lighting a fire.
- William Yeats
On education

An education isn’t how much you have committed to memory, or even how much you know. It’s being able to differentiate between what you know and what you don’t.

- Anatole France

The best way to have a good idea is to have lots of ideas.

- Linus Pauling

If there is no struggle, there is no progress

- Frederick Douglass

The ability to quote is a serviceable substitute for wit.

- W. Somerset Maugham

The selection problem

- In a list of N ordered items, find the kth smallest
  - Highest salary, median, 90%, ...
  - Solve the problem, then solve it efficiently

- Suppose we can re-arrange the items

  Where is kth smallest?
  - It’s x
  - It’s before x
  - It’s after x

The Selection Problem

- In a list of N ordered items, find the kth smallest
  - Highest salary, median, 90%, ...
  - Solve the problem, then solve it efficiently

```cpp
double Select(vector<double> & a, int left, int right, int k)
// pre: left <= right
// post: return k-th smallest in a[left..right]
{
    if (left == right) return a[left];
    int p = Partition(a, left, right); // re-arrange a
    int size = p - left + 1; // left sublist
    if (k <= size) return Select(a, left, p, k);
    else return Select(a, p+1, right, k-size);
}
```

Partition, the picture and invariant

The desired state, where we want to go

The initial state

The intermediate state, and invariant
**Partition: the code**

```c
int Partition(tvector<int> & a, int left, int right)
// pre: left <= right, and legal indices for a
// post: return index and re-arrange elements in a
// so a[left..index] <= a[index+1..right]
{
    int p = left;
    int k;
    for(k=left+1; k <= right; k++)
        if (a[k] <= a[left])
            p++; Swap(a[k], a[p]);
    Swap(a[left], a[p]); return p;
}
```

**QuickSort, the code**

```c
void QuickSort(tvector<int> & a, int left, int right)
// pre: left <= right, and legal indices for a
// post: a[left] <= … <= a[right]
{
    if (left <= right)
        { int p = Partition(a, left, right);
          QuickSort(a, left, p-1);
          QuickSort(a, p+1, right); }
}
```

Why is this fast? When is it slow?
- Invented in 1960, hard to understand, why?
- Usable in mission critical applications? **Introsort** in 1998

**Practice with invariants**

- Remove zeros from an array, leave order of non-zeros unchanged (AP exam, 1987)

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Sketch/write a solution
  - Make it run, make it right, make it fast

<table>
<thead>
<tr>
<th>non-zeros</th>
<th>(ignored)</th>
<th>?</th>
<th>?</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>lnz</td>
<td>k</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Laws governing computer science**

- **Moore’s Law** (1965)
  - The number of transistors per area on a chip double every 18 months
  - Density of transistors => more functionality and speed
- **How about multiple computers?**
- **Amdahl’s Law** (1967)
  - Given: fraction (s) of work to be done is serial (i.e. isn’t parallelizable)
  - Maximum speedup with infinite number of processors is 1/s
What are computers for?

- **Simulation**
- **Communication among people**
  - Storage = communication across time
- **Control**
  - Get physical
  - Get real (time)
  - Get mobile

Application

- **Simulation**
  - Models of the real world (e.g. planets, cities, molecules)
- **Communication among people**
  - Information at your fingertips
  - Telepresence
  - Home
- **Control**
  - Robots
  - Software agents

What’s next

- **CPS 100**
  - Data structures
  - Efficiency
  - Larger programs
- **After that?**
  - Anything!