CS 105 TA Lecture Notes
September 29, 2014

Review of stuff that's been going wrong in the labs

1) Which conditional does **not** contain an error in the code?

- A) `if (a = 50)` Need to use "==" for conditionals
- B) `if (x =! 40)` The correct operator is "`!=`"
- C) `if (c>4 || c<2)` This is the only one that is correct
- D) `if (x > 20 & > 30)` Need to use "`&&`" and repeat `x` for `x > 30`
- E) `IF (y <= 50)` The keyword "`if`" must be all lowercase

2) `var a = [ [2, 3], [4, 5], [6, 2] ];`
What is the value of `a[1][1]`?

- A) 2
- B) 3
- C) 4
- D) 5
- E) 6

Searching

- Random searching:
  - By the time we get through half the array we "expect" to find the value.
  - But we found it at the 6th trail out of 8 doors.

- Halving:
  - In the example we chose the middle number (4th door) and we got 17, but the number is 22. So, we need to find 22 only in doors to the left.
  - The next middle door, i.e. the 6th door, and we got 33. This made it clear that the 5th door must be 22 as the number 22 is between 17 and 33.

A code example of implementing a linear search function:

```
function linearSearch(s, list)
{
      for(var i = 0; i< list.length; i++)
      {
            if(s == list[i])
            {
                  return true;
            }
      }

      return false;
}
```

- Comments:
  - Move sequentially through an array and search one by one.
  - This causes the search time to vary linearly with the number of search items.

- In contrast, for binary search:
  - You need the elements arranged in an order and then use that order information in order to eliminate half of the list completely, and hence reduce the computational time. The array NEEDS TO BE SORTED.
  - So if you double the problem you only need one more comparison.

Sorting

- Fact: Searching is easy relative to sorting. Sorting allows you to search very quickly using binary search.

- Selection sort:
  - Select the first element and check if it's the smallest quantity in the array; if it isn't, replace that first element with the actual lowest element.

For more detailed information on searching and sorting, please refer to the lecture slides.