Lab 11 - wb 28 /01/14

This week we are looking at some basic search algorithms. These are some good ways to manipulate the data in arrays that you are working with. Searching is used in many programs and databases and needs to be as fast as possible to prevent the program from slowing down.

Linear Search

First we are going to look at linear search, this is one of the simplest searches we can do. I am going to search an array of strings for a piece of text and then output the location that it is stored at.

```
#include <string>

using namespace std;

int Linsearch(string valToFind, string vals[], int arrSize)
{
    for(int i=0; i< arrSize; ++i)
    {
        if((vals[i].compare(valToFind))==0) return i; //When comparing strings we cant use the == operator
                                                      //So we use the .compare operator
    }

    return -1;
}

int main()
{
    string myArr[5]={"hey","hello","bye","yes","happy"}; // lets make our array

    int result=-1; // initailise a varaiable to store the result
    string searchval;

    cout<<"What would you like to search for? : ";
    cin>>searchval;

    result = Linsearch(searchval,myArr,5);

    if(result == -1) cout<<"That string does not exist!";
    else
    cout<<"That string is located at array index "<<result<<"!";

    cin.ignore();
    cin.get();
    return 0;
}
```

Binary Search

If our array is sorted then we can use a binary search that works on the divide and conquer method. To do this I am going to use and array of sorted numbers and a recursive function.

```
#include <iostream>
#include <string>

using namespace std;

int binSearch(int valToFind, int vals[],int start, int end)
{
    if(end<start) return -1; //if the full array has been searched then stop
    else
    {
        //now we want to find the mid point
        int mid = ((end-start)/2) +start;

        if(vals[mid] >valToFind) //then search the lower half
            return binSearch(valToFind, vals, start, mid-1);
        else if(vals[mid]< valToFind) // then search the upper half
            return binSearch(valToFind, vals, mid+1, end);
        else //return the value position if found
            return mid;

    }
}

int main()
{
    int myArr[5]={1,4,6,7,9}; // lets make our array

    int result=-1; // initailise a varaiable to store the result
    int searchval;

    cout<<"What would you like to search for? : ";
    cin>>searchval;

    result = binSearch(searchval,myArr,0,4);

    if(result == -1) cout<<"That int does not exist!";
    else
    cout<<"That int is located at array index "<<result<<"!";

    cin.ignore();
    cin.get();
    return 0;
}
```