Handout 9  

Recursion, Binary Search

1. **Recursion** – calling a method from within itself.  
   A way of implementing repetition.

Consider a few examples

1. Given the following method definition

   ```java
   public static void countdown (int x)
   {
     System.out.println(x);
     if (x > 0)
       countdown (x-1);
     else
       System.out.println("STOP");
   }
   ```

   What will be printed when the following call is executed?
   ```java
   countdown(3);
   ```

2. What would method `showArray` do

   ```java
   public static int showArray (int [] arr, int index)
   {
     if (index <= arr.length - 1)
     {
       System.out.print(arr[index] + "");
       showArray (arr, index+1);
     }
   }
   ```

   when the following is executed:

   ```java
   int [] a = {1, 3, 2, 4, 8}
   showArray(a, 2);
   ```
3. What would method `add` do

```java
public static int add (int [] arr, int i)
{
    int sum = 0;
    if ( i <= arr.length - 1 )
        sum = arr[i] + add(arr, i+1);
    else
        System.out.println("STOP");
    return sum;
}
```

when the following is executed:

```java
int [] a = {1, 3, 2, 4, 8}
System.out.println (add (a, 2));
```

4. Binary Search

Searching a list for a particular value is a very common problem. Searching is a thoroughly studied topic. Sequential and binary are two common search algorithms. *Sequential search*: inefficient, but easy to understand and program. *Binary search*: more efficient than sequential, but it only works if *the list is sorted first!*

**Binary search strategy: searching for `target`**

- compare the middle element of the searched portion of the array to the `target`
- if the middle element is equal to value – stop.
- if the middle element is greater– perform Binary Search on the left half of the array
- if the middle element is lower– perform Binary Search on the right half of the array

Notice that each time the value is compared to an element from the the array and they are not equal, *one half of the list* is eliminated.
Method Bsearch (int[] a, int target, int first, int last)
  implements a binary search algorithm

  Searches for target between first and last positions in the array a.
  Returns the index of the entry if the target value is found or -1 if
  it is not found.

  */
  public static int Bsearch(int[] a, int target, int first, int last)
  {

    int result = -1;
    int mid;
    if (first > last)
      result = -1;
    else
    {
      mid = (first + last)/2;

      if (target == a[mid])
        result = mid;
      else if (target < a[mid])
        result = Bsearch(target, first, mid - 1);
      else // (target > a[mid])
        result = Bsearch(target, mid + 1, last);
    }

    return result;
  }