ALGONQUIN COLLEGE

CST8130 – Data Structures

September 12, 2017

Professor : Dr. Anu Thomas
Email: thomasa@algonquincollege.com
Office: T314

Recursion

Recursion - Definition

- Algorithm whose solution calls itself
- Many problems can have both iterative and recursive solutions
- For a problem to be successfully solved recursively, each call must either solve a part of the problem, or reduce the size of the problem
- Recursive solution often involves more overhead (memory)

When to use recursion?

Use recursion if

- The solution naturally suits itself to recursion
- The recursive solution is shorter and more understandable
- The recursive solution runs in acceptable time and space limits

Example - Iteration

Print the numbers from 1 to 5

Iterative Solution – count up:

```
for (int i = 1; i <= 5; i++)
    System.out.println (i);
```

Iterative Solution – count down:

```
for (int i = 5; i >= 1; i--)
    System.out.println (i);
```

Example – Recursion 1

Print the numbers from 1 to 5

Recursive Solution – count down:

```
public static void countDown (int n) {
    System.out.println (n);
    if (n > 0)
        countDown (n-1);
}
 public static void main (String [] args) {
     countDown (5);
}
```

Example – Recursion 2

Print the numbers from 1 to 5

Recursive Solution – count up:

```
public static void countUp (int n) {
    if (n > 0)
        countUp (n-1);
    System.out.println (n);
}
 public static void main (String [] argS) {
     countUp (5);
}
```

Sample 1 – what does this do?

```
public static void recurse (int n) {
    if (n >= 0)  {
        System.out.println("before");
        recurse (n-1);
        System.out.println ("after" + (n-1));
    }
}
 public static void main (String [] argS) {
    recurse (3);
}
```

Sample 2 – what does this do?

```
public static int recurse(int n) {
        if (n > 0)
                return (n * recurse(n - 1));
        return 1;

}

public static void main(String[] argS) {
        System.out.println(recurse(5));
}
```

Questions?