Department of Computer Science
Faculty of Physical Sciences
Ahmadu Bello University, Zaria

COSC 211 : Object Oriented Programming I - LAB10

Objectives:

- Learn to think recursively
- Learn how to write simple recursive methods

1. Thinking Recursively

It is natural to think of task that involves repetition in terms of loops. For example that task of computing the factorial of a number n, could be thought of as finding cumulative product of numbers from 1 to n. That is using the formula"

n! = 1 * . . .* n or 1 if n = 0.

Another approach to solving problems that involves repetition is **divide-and-conquer** approach or **recursion**. In this approach, the solution is defined in terms of a smaller version of the problem. This simplification of the problem is repeated until the smallest form of the problems is reached for which the solution is obvious. For example, the factorial of n could be defined as:

$$n! = \begin{cases} 1 & if\ n = 0 \\ n(n-1)! & if\ n > 0 \end{cases}$$

Some examples of problems that can be thought of in this way are described below:

$$sumFromOneTo(n) = \begin{cases} 0 & if\ n <= 1 \\ sumFromOneTo(n-1) + n & if\ n > 0 \end{cases}$$

$$arrayMinimum(array,\ start) = \begin{cases} array[start] & if\ start = length - 1 \\ \min(array[start], arrayMinimum(array, start+1)) & if\ start < length - 1 \end{cases}$$

$$isMember(element, array, start) = \begin{cases} false & if\ \ start >= length \\ true & if\ \ array[start] == element \\ isMember(eleement, array, start+1), & if\ array[start] != element \end{cases}$$

$$printRange(start, stop, step) = \begin{cases} print\ \ nothing & if\ start > stop \\ print\ start;\ \ printRange(start + step,\ stop, step) & if\ start <= stop \end{cases}$$

2. Writing recursive methods in Java

One of the beauty of recursive algorithms is that they can be implemented almost directly in Java. The code is generally shorter than the loop version and it usually requires lass number of variables and assignments. The following examples implements the algorithms described above.

Example 1: Computes the sum of integers from 1 to a given number, n.

Recursive

```
import java.io.*;


public class SumFromOneTo {

    public static void main(String args[]) {

        System.out.println(sumFromOneTo(10));

    }


    public static int sumFromOneTo(int n) {

        if (n <= 0)

           return 0;

        else

          return sumFromOneTo(n-1)+n;

    }

}
```

Iterative

```
import java.io.*;


public class IterativeSumFromOneTo {

    public static void main(String args[]) {

        System.out.println(sumFromOneTo(10));

    }


    public static int sumFromOneTo(int n) {

       int sum=0;

       for (int i=1; i<=n; i++)

           sum+=i;

       return sum;

    }

}
```

Example 2: Find the minimum element from an array.

Recursive

```
import java.util.Scanner;


public class ArrayMinimum {

   public static void main(String[] args)  {

       Scanner stdin = new Scanner(System.in);


       double[] number = new double[10];

       for(int k = 0; k < number.length; k++)       {

         System.out.print("Please enter element#" + (k + 1)+": ");

         number[k] = stdin.nextDouble();

       }


       System.out.println("The minimum value in the array is " + arrayMinimum(number, 0));

       System.out.println();

   }


   public static double arrayMinimum(double[] x, int start) {

       if (start == x.length-1)

          return x[x.length-1];

       else

          return Math.min(x[start], arrayMinimum(x, start+1));

   }

}
```

Iterative

```
   public static double arrayMinimum(double[] x) {

       double min = x[0];

       for (int i=1; i<x.length; i++)
```

```
        if (x[i] < min)
           min = x[i];
      return min;
   }
```

Example 3: Check if an element is contained in an array.

Recursive

```
import java.util.Scanner;

public class Member {

   public static void main(String[] args)  {

       Scanner stdin = new Scanner(System.in);


       double[] number = new double[10];

       for(int k = 0; k < number.length; k++)       {

         System.out.print("Please enter element#" + (k + 1)+": ");

         number[k] = stdin.nextDouble();

       }


       System.out.print("\nNow enter element to search for: ");

       double element = stdin.nextDouble();


       if (isMember(element, number, 0))

           System.out.println(element+ " is contained in the array");

       else

           System.out.println(element+ " is NOT contained in the array");

   }


   public static boolean isMember(double e, double[] x, int start) {

       if(start >= x.length)

          return false;

       else if (e == x[start])

          return true;

       else

          return isMember(e, x, start+1);

   }

}
```

Iterative

```
   public static boolean isMember(double e, double[] x) {

       for (int i=0; i<x.length; i++)

          if (x[i] == e)

             return true;


       return false;

   }
```

Example 4: Print numbers from a given starting value to a given stopping value incrementing using a given stepping value.

Recursive

```
import java.util.Scanner;
```

```
public class PrintRange {

    public static void main(String args[]) throws IOException {

       Scanner stdin = new Scanner(System.in);

       System.out.print("Enter starting value: ");

       int start = stdin.nextInt();

       System.out.print("Enter stoping value: ");

       int stop = stdin.nextInt();

       System.out.print("Enter stepping value: ");

       int step = stdin.nextInt();

       printRange(start, stop, step);

    }

    public static void printRange(int start, int stop, int step) {

        if (start <= stop) {

           System.out.println(start);

           printRange(start+step, stop, step);

        }

    }

}
```

Iterative

```
    public static void printRange(int start, int stop, int step) {

        for (int i=start; i<stop; i+=step)

           System.out.println(i);

    }
```

4. Assignments

1. Write a recursive method that returns the sum of elements in an array of double. Test your method by writing a main method that creates an array of double of size 10, initialize it by reading data from the user and calling your method to compute the sum and print it. See fig 1 below:

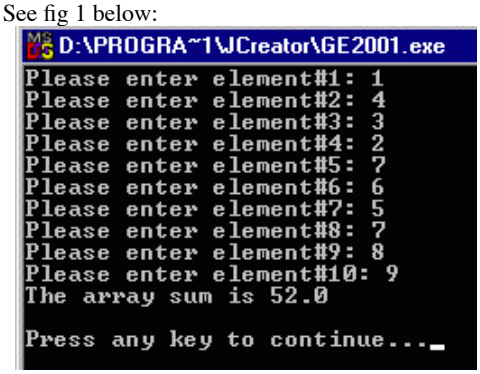


fig 1

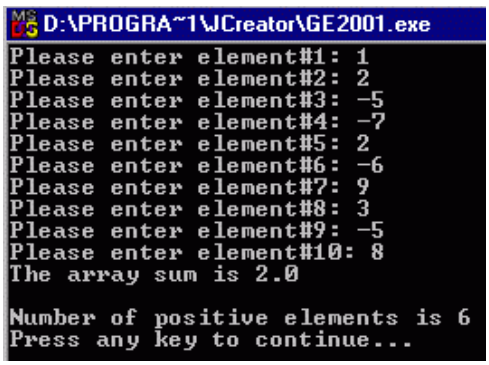
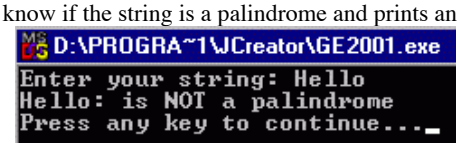


fig 2

2. Improve your program in 1 above by writing another recursive method that counts the number of positive elements in the array. add a statement inside the main method to call this method so that you program prints both the sum and the number of positive elements in the array. See figure 2 above.

3. Write a recursive method that returns true if a string passed to it as parameter is a palindrome (palindrome is a string that reads the same both forward and backward). Test your method by writing a main method that reads a string from the user and then calls your method to know if the string is a palindrome and prints an appropriate message. See fig 3 below:



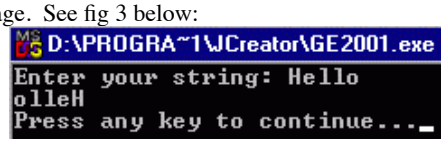


fig 4

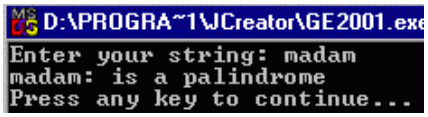


fig 3

4. Write a recursive method that prints a string passed to it as parameter in reverse order. Test your program by writing a main method that reads a string from the user and calls your method to print it in reverse order. See the fig 4. above.