German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Wael Abouelsaadat
Dr. Mohammed Abdel Megeed

Introduction to Computer Programming, Spring Term 2017
Practice Assignment 5

Discussion: 18.3.2017 - 23.3.2017

Exercise 5-1 Blast Off

Write a recursive method `countdown` that takes a single integer as a parameter from the user and prints the numbers from `n` until `1` and then prints "`Blastoff!`". If the parameter is zero, it prints only the word "`Blastoff!`"

For example if the user will enter `6` then the program should output:

```
6
5
4
3
2
1
Blastoff!
```

Solution:

```java
import java.util.*;
public class Countdown {
        public static void countdown (int n) {
                if (n == 0) {
                        System.out.println ("Blastoff!");
                } else {
                        System.out.println(n);
                        countdown (n-1);
                }
        }
        public static void main(String [] args)
        {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please enter a number: ");
                int input = sc.nextInt();
                countdown(input);
        }
}
```

Exercise 5-2 Power

Consider the evaluation of x^n, where n is a non-negative integer. Write a recursive method `powerRec` to calculate x^n.

Write a `main` method to test your method.

Solution:

```
public static int PowerRec(int x,int n){
        if (n == 0)
                return 1;
        else {
                if ( n%2 == 0)
                        return ( PowerRec(x, n/2) * PowerRec(x,n/2) );
                else
                        return ( x * PowerRec(x, n-1) );
        }
}
```

Exercise 5-3 Natural Logarithm

Write a recursive method `constantRec` to calculate the value of the mathematical constant e which is defined as:

$$e(n) = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

Implement first a recursive method `factorial`, where $factorial(n) = n!$.

Write a `main` method to test your method.

Solution:

```
public class ConsR {
        public static double constantRec(int n) {
                if ( n == 0 )
                        return 1.0;
                else
                        return (1.0 / fact(n)) + constantRec(n-1);
        }

        public static double fact(double n) {
                if (n==0)
                        return 1;
                else {
                        return n * fact(n-1);
                }
        }
        public static void main (String [] args){
                System.out.println(constantRec(8));
                System.out.println(constantRec(15));
        }
}
```

Exercise 5-4 MultilplyRec
To be discussed in the Lab

Write a recursive method `multiplyRec` that perform the multiplication of two numbers.

$$f(x, y) = x * y$$

Your method will use only the addition and subtraction operators.

Write a `main` method to test your method.

Hint:

$$x * y = \underbrace{x + x + \ldots + x}_{y \text{ times}}$$

Solution:

```
import java.util.*;
public class MultiplyRec{
        public static int MultiplyRec(int num1, int num2){
                if (num2 == 0)
                        return 0;
                else
                        return (num1 + MultiplyRec(num1, num2-1));
        }
        public static void main(String args[]){
                Scanner sc = new Scanner(System.in);
                System.out.print("Enter␣the␣first␣num:␣");
                int numA = sc.nextInt();
                System.out.print("Enter␣the␣second␣num:");
                int numB = sc.nextInt();
                System.out.println("The␣product␣is␣" + MultiplyRec(numA,
                   numB));
        }
}
```

Exercise 5-5 Division

Write a recursive Java method `divideRec` that perform the integer division operation of two numbers. If y is a negative number, an error message has to be displayed.

$$f(x, y) = \frac{x}{y}$$

DO NOT USE THE JAVA PREDEFINED OPERATOR `/`.

Hint:

$$\frac{x}{y} = \begin{cases} 0 & : \quad \text{if } x < y \\ 1 + \frac{x-y}{y} & : \quad \textit{Otherwise} \end{cases}$$

Write a `main` method to test your method.

Solution:

```
public static int divideRec(int x, int y) {
        if (y <= 0)
                return -1;
        else if (x < y)
                return 0;
        else
                return 1 + divideRec(x-y, y);
}
```

Exercise 5-6 Modulus

Write a recursive Java method `ModulusRec` that perform the modulus operation of two integers. If y is a negative number, an error message has to be displayed.

$$f(x,y) = x\%y$$

DO NOT USE THE JAVA PREDEFINED OPERATOR `%`.

Hint:

$$x\%y = \begin{cases} 0 & : \quad \text{if } x = 0 \\ x & : \quad \text{if } x < y \\ (x-y)\%y & : \quad \textit{Otherwise} \end{cases}$$

Write a `main` method to test your method.

Solution:

```
public class ModulusRec{
        public static int modRec(int x, int y){
                if (x == 0)
                        return 0;
                else if (x < y)
                        return x;
                else
                        return modRec(x-y, y);
        }
        public static void main(String [] args) {
                System.out.println(modRec(7,2));   // where 7%2=1
                System.out.println(modRec(0,2));   // where 0%2=0
                System.out.println(modRec(3,5));   // where 3%5=3
                System.out.println(modRec(6,2));   // where 6%2=0
        }
}
```

Exercise 5-7 Sum of Digits

Write a recursive method to determine the sum of the digits of an integer. For example, the sum of digits of `51624` is `5 + 1 + 6 + 2 + 4 = 18`.

Solution:

```
import java.util.*;
public class SumOfDigits {
        public static int add(int num)
        {
                if(num < 10)
                        return num;
                else
                        return ((num % 10) + add(num / 10));
        }
        public static void main(String [] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please␣enter␣a␣number:␣");
                int input = sc.nextInt();
                System.out.println("The␣sum␣of␣the␣digits␣of␣" + input + "␣
                    is␣"
                                                    + add(input));
        }
}
```

Exercise 5-8 Number of Digits

Write a recursive Java method `numberDigitsRec`, which given an integer, returns its number of digits. For example the call `numberDigitsRec(12312)` will return `5`.

Solution:

```
public static int numberDigitsRec(int x){
        if (x%10 == x)
                return 1;
        else
                return numberDigitsRec(x/10) + 1;
}
```

Exercise 5-9 Prime

A prime number is an integer that cannot be divided by any integer other than one and itself. For example, 7 is prime because its only divisors are 1 and 7. The integer 8 is not prime because its divisors are 1, 2, 4, and 8.

Another way to define prime is:

```
prime(N)     = prime(N, N-1)

prime(N, 1) = true

prime(N, D) = false          if D divides N
              prime(N, D-1)  otherwise
```

For example,

```
prime(4)   = prime(4,3)
prime(4,3) = prime(4,2)
prime(4,2) = false
```

Another example,

```
prime(7)   = prime(7,6)
prime(7,6) = prime(7,5)
prime(7,5) = prime(7,4)
prime(7,4) = prime(7,3)
prime(7,3) = prime(7,2)
prime(7,1) = true
```

Translate the math-like definition of `prime` into two Java methods that return boolean. Use the `%` operator to test divisibility. Put your method into a class, write a testing class, and test your program. (Look at Triangle.java in this assignment.)

Solution:

```
import java.util.*;
public class Prime {
        public static boolean Prime(int n) {
                if (n == 1)
                        return true;
                else
                        return prime(n, n-1);
        }
        public static boolean prime(int n, int d) {
                if (d == 1)
                        return true;
                else if (n%d == 0)
                        return false;
                else return prime(n, d-1);
        }

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please␣enter␣a␣positive␣number");
                int n = sc.nextInt();
                boolean result = prime(n);
                if (result)
                        System.out.println("The␣number␣" + n + "␣is␣prime."
                            );
                else
                        System.out.println("The␣number␣" + n + "␣is␣not␣
                            prime.");
        }
}
```

Exercise 5-10 Cube numbers

Write a program that implements this definition of cube numbers, where `N` is an integer entered by the user.

```
cube(1) = 1
cube(N) = cube(N-1) + 3(square(N)) - 3N + 1
```

Implement the `square()` method using this definition (also given in Lab assigment 7):

$$(N-1)^2 = N^2 - 2N + 1$$

Solution:

```
import java.util.*;
public class Cube {
        public static int cube (int n) {
                if (n == 1)
                        return n;
                else
                        return cube(n-1) + (3 * square(n)) - (3 * n) + 1;
        }
        public static int square (int n) {
                if (n == 1)
                        return n;
                else
                        return square(n - 1) + (2 * n) - 1;
        }

        public static void main(String args[]) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please␣enter␣a␣number:␣");
                int input = sc.nextInt();
                System.out.println("The␣cube␣number␣of␣" + input + "␣is␣␣"
                    + cube(input));
        }
}
```

Exercise 5-11 Binomial Coefficient

The binomial coefficient $\binom{n}{k}$ is the number of ways of picking k unordered outcomes from n possibilities, also known as a combination or combinatorial number.

The binomial coefficient is defined recursively as follows:

$$\binom{n}{k} = \begin{cases} 1 & : \text{ if } k = 0 \\ 1 & : \text{ if } n = k \\ \binom{n-1}{k} + \binom{n-1}{k-1} & : \text{ otherwise} \end{cases}$$

Write a recursive method to calculate the binomial coefficient. Make sure that n is less than k.

Write a `main` method that will allow the user to enter the actual parameters of the binomial coefficient method. Use either `Scanner` class or the command line arguments as input mechanism.

```
public static void main(String[] args)  {
        Scanner sc = new Scanner(System.in);
        ....
```

Solution:

```
public class Binom {
        public static long choose(int n, int k)
        {
                long answer;              // Hold the answer
                if(k == 0 || n == k)
                        answer = 1;
                else
                        answer = choose(n-1, k) + choose(n-1, k-1);
                return answer;
        }

        public static void main(String args[]) {
                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();             // size of the set
                int k = sc.nextInt();             // number of objects
                System.out.println("There␣are␣" + choose(n, k) + "␣ways␣to␣
                    choose␣"
                                                                  + k + "␣
                                                                      objects␣
                                                                      out␣of␣"
                                                                       + n);
        }
}
```

Exercise 5-12 CountRec

Write a recursive method named `countRec` that accepts two arguments: a `String` value, and a `char` value. Your method is to return the total number of times the character appears inside of the string.

Solution:

```
public class Count {
        public static int count(String x, char c) {
                if (x.length() == 0)
                        return 0;
                else if (x.charAt(0) == c)
                        return count(x.substring(1), c) + 1;
                else
                        return count(x.substring(1), c);
        }
        public static void main(String args[]) {
                System.out.println( count("Hello", ’l’) ); // displays 2
                System.out.println( count("Hello", ’o’) ); // displays 1
                System.out.println( count("Hello", ’H’) ); // displays 1
                System.out.println( count("Hello", ’h’) ); // displays 0
        }
}
```

Exercise 5-13 Reverse

Consider reversing the characters in a string. Write a recursive Java method `reverseRec` that returns a new string with the same characters of the original string, but in reversed order.

Think about the base case and recursive case:

- Base case: `ReverseRec("") => ""`
- Recursive case:

  `reverse("ABCDE") => "E" + ReverseRec("ABCD") => ......  => "EDCBA"`

Hint: In addition to the methods `charAt` and `length`, use the predefined method `substring()`. For example if we have a string `String s = "CSEN202"`, then `s.substring(1)` returns `"SEN202"`, i.e. the string `s` without the first character.

Write a `main` method to test your method.

Solution:

```
public class Reverse {
        public static String reverseRec(String s) {
                if(s.length() <= 1)
                        return s;           // 1 or 0 character string is
                            its own reverse
                else
                        return reverseRec(s.substring(1)) + s.charAt(0);
        }
        public static void main(String [] args) {
                String  x = ("abcdefghij");    // Sample string
                System.out.println (reverseRec(x));
        }
}
```

Exercise 5-14 Palindrome
To be discussed in the Tutorial

Write a recursive method `palindrome` that takes a single string as a parameter from the user and returns whether the string is palindrome or not.

Solution:

```
import java.util.Scanner;
public class PaliRec{

        public static void main (String[] args)
        {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please␣enter␣a␣word␣to␣be␣checked␣for␣palindrome");
                String x = sc.nextLine();
                System.out.println(palindrome(x));
        }

        public static boolean palindrome(String x)
        {
                if(x.length()==0 || x.length()==1 )
                        return true;
                if(x.charAt(0) == x.charAt(x.length()-1) )
                        return palindrome(x.substring(1,x.length()-1));

                return false;
        }

}
```

Exercise 5-15 Replace

Write a recursive method **replace** that takes two arguments: `String` and `char` and replaces each occurrence of the character with a **'*'**

Solution:

```
public class Replace{
        public static String replace(String s, char c) {
                if(s.length() == 0) {
                        return "";
                } else if(s.charAt(0) != c) {
                        return s.charAt(0) + replace(s.substring(1), c);
                } else {
                        return "*" + replace(s.substring(1), c);
                }
        }
        public static void main(String [] args) {
                System.out.println(replace("Computer␣Science",'e'));
        }
}
```

Exercise 5-16 Eliminate

Write a recursive method **eliminate** that takes a `String` and a `char` and deletes each occurrence of this character from the string.

Solution:

```
public static String elim(String s, char c)
{
        if(s.length() == 0)
                return "";
        else if(s.charAt(0) != c)
                return s.charAt(0) + elim(s.substring(1), c);
        else
                return elim(s.substring(1), c);
}
```

Exercise 5-17 Recursion Tracing 1

Give the following program:

```
public static void mystery1(int a, int b){
        if (a <= b){
                int m =(a + b) / 2;
                System.out.print(m + "␣");
                mystery1(a, m - 1);
                mystery1(m+1, b);
        }
}
```

What is the output of the main method below? Justify your answer with a tracing table.

```
public static void main (String [] args){
        int n = 6;
        mystery1(0, n);
}
```

Solution:

```
3 1 0 2 5 4 6
```

Exercise 5-18 Recursion Tracing 2
To be discussed in the tutorials

Give the following program:

```
public static void mystery(String prefix, String remaining, int k){
        if(k == 0){
                System.out.println(prefix);
                return;
        }
        if (remaining.length() == 0) return;
        mystery(prefix + remaining.charAt(0), remaining.substring(1), k-1);
        mystery(prefix, remaining.substring(1), k);
}
```

a) What is the value returned by the following invocation? Trace your program. `mystery("", "CSEN", 3)`

Solution:

```
CSE
CSN
CEN
SEN
```

b) What does the above method do? Give an concise verbal description of how the value returned by `mystery("", s, k)` is related to the values of the parameters `s` and `k`.

Solution:

The method prints out all subsequences of `s` of length `k`.

Exercise 5-19 Search

Write a recursive method `search()` to search for a `char` inside a `String` and returns its position inside the `String`. The method should returns -1 if the `char` is not in the `String`.
Use the following **main** method to test your program:

```
public static void main(String[] args) {
        System.out.println(search("example",'a'));
}
```

Hint:

- “Hello”.substring(1) returns “ello”.
- “Hello”.substring(1,4) returns “ell”.
- “Hello”.substring(0,s.length()-1) returns “Hell”.

Solution:

```
public static int search(String s,char c)
{
        if(s.length()==0)
                return -1;
        else if(s.charAt(s.length()-1) == c)
                return s.length() - 1;
        else
                return search(s.substring(0, s.length()-1), c);
}
```

Exercise 5-20 Put At Front - Final Spring 2013
To be discussed in the Lab

Write a recursive method `putAtFront` that takes two parameters, a string s and a character c. The method returns a string with all occurrences of c placed at the front of the string and all other characters afterwards, in the same order they appear in the input string. If c does not exist, then the output string is the same as s. If c is at the beginning of s, and it does not appear in s any more time, then the output string is also the same as s.

The following list illustrates 4 different calls and the correct return value from calling the method each time.

```
Call: putAtFront("sce", 'c');
Return: cse
// note how c is at the front of the returned
// string and the remaining characters afterwards.
Call: putAtFront("static", 't');
Return: ttsaic
// here t appears twice in input string s.
// In the returned string, both are at front,
// and the remaining afterwards.
Call: putAtFront("banana", 'a');
Return: aaabnn
Call: putAtFront("java", 'j');
Return: java
Call: putAtFront("ALL", 'L');
Return: LLA
```

Solution:

```
import java.util.Scanner;

public class PutAtFront{

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please enter  the string: ");
                String s = sc.nextLine();
                System.out.println("Please enter  the character: ");
                char c = sc.nextLine().charAt(0);
                System.out.println(
                        "The generated string with the occurences of "
                        + c + " put at front is: " + putAtFront(s,c));
        }

        public static String putAtFront(String s, char c) {
                if (s.length() == 0)
                  return s;

                else {
                    if (s.charAt(s.length() - 1) == c) {
                        return c + putAtFront(s.substring(0,
                        s.length() - 1), c);
                        }
                        else {
                                return putAtFront(s.substring(0,
                                s.length() - 1), c) + s.charAt(s.length() -
                                    1);
                        }
                }
        }
}
```

Exercise 5-21 PerfectRec

A positive integer is said to be perfect if the sum of its factors (excluding the integer itself) is that integer. For example, 6 is perfect, since the numbers that divide into it exactly are 1, 2, 3, and 6, and the sum of 1, 2, and 3 is itself 6.

Write a recursive method `perfectRec` to calculate the sum of divisors of `n` and use it to output whether `n` is perfect or not.

Solution:

```
import java.util.Scanner;
public class PerfectRec2 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please enter  a number: ");
                int query = sc.nextInt();

            int sum = sumFactorsTo(query, query-1);

            if(sum == query) {
                System.out.println(query + " is perfect");
           } else {
              System.out.println(query + " isn't perfect: The sum is " +
                sum);
        }
    }

    public static int sumFactorsTo(int num, int max) {
        if(max == 0) {
            return 0;
        } else {
          int sub = sumFactorsTo(num, max - 1);
            if(num % max == 0) {
                sub += max;
            }
            return sub;
        }
    }

      /* Another solution

          public static int sumFactorsTo(int num, int index) {
        if(index == num) {
            return 0;
        } else {
              int sumAfter = sumFactorsTo(num,index+1);
           if(!(num % index == 0)) {
              return sumAfter;
           }
           return index+ sumAfter;
          }
    }

      */

}
```

Exercise 5-22 Look And Say

The look-and-say sequence is the sequence of integers beginning as follows:

`1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, ...`

To generate a member of the sequence from the previous member, read off the digits of the previous member, counting the number of digits in groups of the same digit. For example:

- 1 is read off as “one one” or 11.
- 11 is read off as “two ones” or 21.
- 21 is read off as “one two, then one one” or 1211.
- 1211 is read off as “one one, then one two, then two ones or 111221.
- 111221 is read off as “three ones, then two twos, then one one” or 312211.

Write a recursive Java method `lookAndSay` and prints the first n^{th} terms of this sequence. **Hint:** you can create a helper method that takes a String as a parameter acting as the i^{th} term of the sequence and returns a String representing the i^{th}+1 term..

Solution:

```
import java.util.Scanner;
public class LookAndSay {
    public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Please␣enter␣a␣number:");
                int query = sc.nextInt();
                        lookAndSay(query,query,"");
    }

    public static void lookAndSay(int n , int m, String current) {
                if(m==0)
                        System.out.print("...");
                else{
                        if(m==n)
                                current = "1";
                        else
                        {
                                if(m<n)
                                        current = helper(current) ;
                        }

                System.out.print(current+",␣");
                lookAndSay(n,m-1, current);
                }


        }


        public static String helper(String x)
        {
                if(x.length()==0)
                        return "";

                char c = x.charAt(0);
                int i;
                int counter =1;
                for(i=1; i<x.length();i++)
                {
                        if(c==x.charAt(i))

                                counter++;
                        else
                                break;
                        }
                return "" + counter+ c+ helper(x.substring(i,x.length()));

        }

}
```

Exercise 5-23 Recursion Tracing 3

Given the following method:

```
public static void mystery( int n) {
    if (n/2 == 0 ) {
            System.out.print(n);
        } else if (n % 2 == 0) {
                System.out.print("(" + n + " + ");
                mystery(n - 1);
                System.out.print(")");
            } else {
                System.out.print("(");
                mystery(n - 1);
                System.out.print(" + " + n + ")");
            }
}
```

a) What does the method display for the following call:

```
mystery(6);
```

Trace your method using a stack.

Solution:

`mystery(6); -> (6 + ((4 + ((2 + 1) + 3)) + 5))`

b) What does the method displays for any integer?

Solution:

The recursive method `mystery` that accepts an `int n` as a parameter and prints out the numbers 1 through `n` inclusive in a particular pattern that looks like a set of mathematical additions wrapped in parentheses. The order of the numbers should begin with all of the evens in downward order, followed by all of the odds upward from 1. Each time a number is added to the pattern, a new set of parentheses and a + sign are added too.

Exercise 5-24 MergeRec

Write a recursive method mergeRec that given two strings displays the characters of the given strings in an alternating way. Note that the two strings could be of different length. Note that you are not allowed to use any additional strings.

Once you execute the following main method

```
    public static void main(String[] args) {
    String a = "hlo";
    String b = "el";
    mergeRec(a,b);
}
```

the following should be displayed:
h e l l o

```
public static void mergeRec(String a,  String b) {
```

Solution:

```
public static void mergeRec(String a,  String b)
{
    helper(a,b,0,0);
}
public static void helper (String a, String b, int i, int j)
{
    if(a.length()==i && b.length()==j)
    {
        return;
    }
    if(b.length()==j)
    {
        System.out.print(a.charAt(i)+" ");
        helper (a,b,++i,j);
    }
    else{
        if(a.length()==i)
        {
            System.out.print(b.charAt(j) + " ");
            helper (a,b,i,++j);
        }
        else
        {
            System.out.print(a.charAt(i) + " " + b.charAt(j) +" ");
            helper(a,b,++i,++j);
        }
    }
}
```