FARICHA AULIA

JOBSHEET 14. Function 2

Purpose

- Students understand the concept of recursive functions
- Students are able to implement recursive functions in program code

Tools and materials

- PC/Laptop
- Browsers
- Internet connection
- Anaconda3 + Java kernel (optional)

Practicum

Experiment 1

In this experiment, a program will be made to calculate the factorial value of a number using a recursive function. In addition, a function will also be made to calculate the factorial value using an iterative algorithm as a comparison

1. Create a static function with the name **recursiveFactorial()**, with the return data type of the function int and having 1 parameter with the data type int in the form of a number whose factorial value will be calculated

```
static int recursiveFactorial(int n){
    if(n==0){
        return (1);
    }
    else {
        return(n*recursiveFactorial(n-1));
    }
}
```

In [3]:
```
// Write down the code for Experiment 1 Step 1
static int recursiveFactorial(int n){
    if(n == 0){
        return (1);
    }
    else {
        return (n*recursiveFactorial(n - 1));
    }
}
```

2. Create another static function with the name **iterativeFactorial()**, with the return data type of the function int and having 1 parameter with the int data type in the form of a number whose factorial value will be calculated.

```
static int iterativeFactorial(int n){
    int factor = 1;
    for(int i=n; i>=1; i--){
        factor = factor * i;
    }
    return factor;
}
```

In [2]:
```
// Write the code for Experiment 1 Step 2
static int iterativeFactorial(int n){
    int factor = 1;
    for (int i = n; i >= 1; i--){
        factor = factor * i;
    }
    return factor;
}
```

3. Make calls to the two functions that have been made previously, and display the results obtained.

```
System.out.println(recursiveFactorial(5));
System.out.println(iterativeFactorial(5));
```

In [4]:
```
// Write down the code for Experiment 1 Step 1, 2, 3
static int recursiveFactorial(int n){
    if(n == 0){
        return (1);
    }
    else {
        return (n*recursiveFactorial(n - 1));
    }
}

static int iterativeFactorial(int n){
    int factor = 1;
    for (int i = n; i >= 1; i--){
        factor = factor * i;
    }
    return factor;
}

System.out.println(recursiveFactorial(5));
System.out.println(iterativeFactorial(5));
```

```
120
120
```

4. When traced to the function call recursiveFactorial(5), the process that occurs can be illustrated as follows:

```
recursiveFactorial(5)  = 5 * recursiveFactorial(4)
                       = 5 * (4 * recursiveFactorial(3))
                       = 5 * (4 * (3 * recursiveFactorial(2)))
                       = 5 * (4 * (3 * (2 * recursiveFactorial(1))))
                       = 5 * (4 * (3 * (2 * (1 * recursiveFactorial(0)))))
-------------------------------------------------------------------------
                       = 5 * (4 * (3 * (2 * (1 * 1))))
                       = 5 * (4 * (3 * (2 * 1)))
                       = 5 * (4 * (3 * 2))
                       = 5 * (4 * 6)
                       = 5 * 24
```

Question

1. What is a recursive function?
2. What is an example of a recursive function use case?
3. In Experiment1, is the result given by the recursiveFactorial() function and the iterativeFactorial() function the same? Explain the difference between the flow of the program in the use of recursive functions and iterative functions!

> 1. It is a function like a loop method without iteration that calls the function itself until the conditions are met.
>
> 2. Used to calculate and find factor values, rank.
>
> 3. Similarly, if the function is recursive, the program will stop if the condition (base case) is true. if function iteratively, the program will stop if the condition is false.

Experiment 2

In this experiment, we will make a program to calculate the power of a number using a recursive function.

1. Create a static function with the name **calculatePower()**, with the return data type of the function int and having 2 parameters with the data type int in the form of a number to be calculated and the power of the number

```
static int calculatePower(int x, int y){
    if(y==0){
        return(1);
    }
    else{
        return(x* calculatePower(x, y-1));
    }
}
```

2. Declare Scanner with name sc
3. Create two variables of type int with the name number and power
4. Add the following code to accept input from the keyboard

```
System.out.print("The calculated number is : ");
number = sc.nextInt();
System.out.print("Power : ");
power = sc.nextInt();
```

5. Call the calculatePower function that was created earlier by sending two parameter values.

```
System.out.println(calculatePower(number,power));
```

In [5]:
```
// Write the code for Experiment 2 Step 1 - 5
static int calculatePower(int x, int y) {
    if (y == 0){
        return (1);
    }
    else {
        return (x * calculatePower(x, y -1));
    }
}

import java.util.Scanner;
Scanner sc = new Scanner(System.in);
int number,power;
System.out.print("The calulated number is: ");
number = sc.nextInt();
System.out.print("Power: ");
power = sc.nextInt();
System.out.println(calculatePower(number,power));
```

```
The calulated number is: 4
Power: 3
64
```

Question

1. In Experiment 2, there is a recursive call to the function calculatePower(number, power) on the main function, then the function is called calculatePower() repeatedly. Explain how long the process of calling the function will run!

> Answer : The program will continue to call the function on the recursion call, and stop executing if the value of y = 0

Experiment 3

In this experiment, a program will be made to calculate the amount of customer money deposited in the bank after earning interest for several years using a recursive function.

1. Create a static function with the name **calculateBankInterest()**, with the data type as the return function double and having 2 parameters with the data type int in the form of customer balance and duration of saving. In this case, the interest determined by the bank is 11% per year. Because the calculation of interest is interest * balance, so to calculate the amount of money after adding interest is balance + interest * balance. In this case, the interest rate is 0.11 * balance, and the balance is considered 1 * balance, so 1 * balance + 0.11 * balance can be condensed to 1.11 * balance for calculating the balance after adding interest (in a year).

```
static double calculateBankInterest(double balance, int year){
    if(year == 0){
        return (balance);
    }
    else{
        return (1.11 * calculateBankInterest(balance, year-1));
    }
}
```

2. Declare Scanner with name sc
3. Create a variable of type double with the name startingBalance and a variable of type int named year
4. Add the following code to accept input from the keyboard

```
System.out.print("Starting balance amount : ");
startingBalance = sc.nextInt();
System.out.print("Saving time (year) : ");
year = sc.nextInt();
```

5. Call the calculated Interest function that was created earlier by passing two parameter values.

```
System.out.print("Amount of money after " + year + " year: ");
System.out.println(calculateBankInterest(startingBalance, year));
```

In [8]:
```
// Write the code for Experiment 3 Step 1 - 5
static double calculateBankInterest(double balance, int year) {
    if (year == 0) {
        return (balance);
    }
    else {
        return (1.11 * calculateBankInterest(balance, year - 1));
    }
}

import java.util.Scanner;
Scanner sc = new Scanner(System.in);
double startingBalance;
int year;
System.out.print("Starting balance amount: ");
startingBalance = sc.nextInt();
System.out.print("Saving time (year): ");
year = sc.nextInt();
System.out.print("Amount of money after " + year + " year: ");
System.out.println(calculateBankInterest(startingBalance, year));
```

```
Starting balance amount: 1000000
Saving time (year): 3
Amount of money after 3 year: 1367631.0000000002
```

Question

1. In Experiment3, state which block of program code is a “base case” and a “recursion call”!

> a. base case if (year == 0) { return (balance); }

> b. recursion call else { return (1.11 * calculateBankInterest(balance, year - 1));

Task

1. Write a program to display numbers n to 0 using recursive and iterative functions. (**Recursive Descending Series**).

In [9]:
```
// Write down the answer to task number 1
static int recursiveDescendingSeries(int n){
    if(n == 0){
        return(0);
    }
    else{
        return (recursiveDescendingSeries(n-1)+1);
    }
}
static int iterativeDescendingSeries(int n){
int i;
for (i = n; i >= 1; i--){
    System.out.print(i);
    System.out.print(" ");
}
return i;
}
System.out.println("Entered value : " + recursiveDescendingSeries(50));
System.out.println(iterativeDescendingSeries(50));
```

```
Entered value : 50
50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 0
```

2. Write a program in which there is a recursive function to calculate factorial numbers. For example f = 8, it will produce 1+2+3+4+5+6+7+8 = 36 (**Recursive Addition**).

In [10]:
```
// Write down the answer to task number 2
int i = 0;
static int recursive(int n) {
    if (i == n)
        return 0;
    else {
        i++;
        System.out.print( i + " + ");
        return(i + recursive(n));
    }
}
System.out.print(" = "+ recursive(8));
```

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 +  = 36
```

3. Write a program in which there is a recursive function to check whether a number n is a prime number or not. n is not a prime number if it is divisible by a number less than n. (**CheckPrimeRecursive**).

In [11]:
```
// Write down the answer to task number 3
import java.util.Scanner;
Scanner sc = new Scanner(System.in);
int n ;
static int checkPrimaRecursive(int p){
    if (p == 1){
         return 1;
    }
    else if(n%p==0){
         return 0;
    }
    else{
    return checkPrimaRecursive(p-1);
    }
}
System.out.print("Enter number: ");
n = sc.nextInt();
if(n>1){
    int p = checkPrimaRecursive(n-1);
    if (p==1){
        printf("%d Primes\n", n);
    }
    else{
        printf("%d Not primes\n", n);
    }
}
    else{
        printf("Not primes\n", n);
    }
```

```
Enter number: 7
7 Primes
```

4. A pair of newly born guinea pigs (male and female) are placed in a breeding. After two months the couple of guinea pigs gave birth to a couple of twin guinea pigs (male and female). Each couple of guinea pigs that are born will also give birth to a couple of guinea pigs every 2 months. How many couple of guinea pigs are there at the end of the 12th month? Write the program using recursive functions! (**Fibonacci**). The following is an illustration in tabular form.

Month-	Number of Couple		Total Couple
	productive	not productive	
1	0	1	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8
7	5	8	13
8	8	13	21
9	13	21	34
10	21	34	55
11	34	55	89
12	55	89	144

In [12]:
```
// Write down the answer to task number 4
import java.util.Scanner;
Scanner in = new Scanner(System.in);

static int couple(int n, int a, int b) {
    if (n == 0) return a;
    if (n == 1) return b;
    return couple(n - 1, b, a + b);
}
printf("Enter month: ");
int month = in.nextInt();
printf("In month %d there %d couple \n\n", month , couple(month, 0, 1));
```

```
Enter month: 4
In month 4 there 3 couple
```