Trường Đại Học Quốc Tế - ĐHQG TP.HCM

# LAB REPORT

Course: Algorithms & Data Structures LAB 5

**Full Name**: Trần Minh Phúc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Student's ID**: ITCSIU24070 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

# Problem 1: Use the following function puzzle(..) to answer problems 1 - 3.

```
int puzzle(int base, int limit)
{ //base and limit are nonnegative numbers
        if ( base > limit )
        return -1;
        else if ( base == limit )
        return 1;
        else
        return base * puzzle(base + 1, limit);
}
```

## Identify the base case(s) of function puzzle(..)

We have already known that the base case define the condition that stops the function from recalling itself again. Therefore, there are two base cases in the given code, which are:

```
if ( base > limit )
return -1;
else if ( base == limit )
return 1;
```

## Identify the recursive case(s) of function puzzle(..)

Since the recursive case recall itself again, we can easily find it in the given code:

```
else
        return base * puzzle(base + 1, limit);
```

## What displayed

a. System.out.println(puzzle(14,10))

   Since $base(14) > limit(10)$ - base case, the prinln() method will display $-1$

b. System.out.println(puzzle(4,7))

Since $base(4) < limit(7)$ - recursive case, the println() method will display 120, derived from a number of calls $(4 * 5 * 6 * 1)$.

c. System.out.println(puzzle(0,0))

The println() method would display 1 as the $base(0) = limit(0)$.

## Problem 2: Complete the Java code to recursively evaluate the sum: sum $= 1 + 1/2 + 1/3 +...+1/n$, $n > 1$.

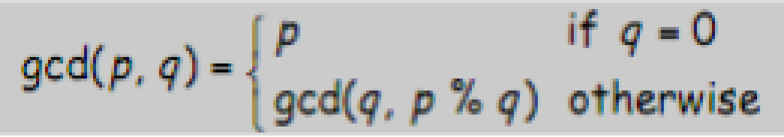
```
double sum(int n)       // n>=1
{
    if(n == 1)
        return 1;
    return 1/n + sum(n-1);
}
```

## Problem 4: Write a recursive function that finds and returns the minimum element in an array, where the array and its size are given as parameters.

```
public class FindMin {
  int findmin(int[] arr, int n) {
      if (arr == null || arr.length == 0) {
        throw newIllegalArgumentException("Array␣must␣not␣be␣null␣or␣empty");
    }
      if (n == 1) {
        return arr[0];
      }
    return min(arr[n - 1], findmin(arr, n - 1));
    }
  public int min(int a, int b) {
      if (a > b) {
```

```java
        return b;
      }
    return a;
  }
}
```

## Problem 6: Write a method that receives two integers and returns the largest common divisor. The formula to calculate the Largest common divisor is shown below:



$$gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ gcd(q, p \% q) & \text{otherwise} \end{cases}$$

```java
public class Gcd {
    public int gcd(int p, int q) {
      if (q == 0) {
        return p;
      }
    return gcd(q, p % q);
    };
}
```

## Problem 8: Write a recursive function to generate all subsets of a given set.

```java
import java.util.ArrayList;
import java.util.List;

  public static List<List<Integer>> subsets(int[] nums) {
    List<List<Integer>> result = new ArrayList<>();
    backtrack(result , new ArrayList<>() , nums , 0);
```

```
      return result;
    }

  private static void backtrack(List<List<Integer>> result, List<Integer>
  tempList,int[] nums, int start) {
    result.add(new ArrayList<>(tempList));
    for (int i = start; i < nums.length; i++{
      tempList.add(nums[i]);
      backtrack(result, tempList, nums, i + 1);
      tempList.remove(tempList.size() - 1);
    }
  }
}
```

## Problem 10: Use recursion to generate a Sierpinski triangle fractal

```
public class SierpinskiRecursive {
public void printSierpinski(int n, int y) {
  if (y < 0) {
    return;
  }
  printSpaces(y);
  printLine(0, n, y);
  System.out.println();
  printSierpinski(n, y - 1);
}

public void printSpaces(int count) {
  if (count == 0) {
    return;
  }
  System.out.print("␣");
  printSpaces(count - 1);
}

public void printLine(int x, int numberOfRows, int y) {
  if (x + y >= numberOfRows) {
        return;
  }
  if ((x & y) != 0) {
        System.out.print("␣␣");
  } else {
        System.out.print("*␣");
  }
  printLine(x + 1, numberOfRows, y);
  }
}
```

*This is the end of the report*