**ALGORITHMS & DATA STRUCTURES 2019-2020**

**Lahcen Ouarbya**

**TOPIC 2: RECURSION**

**WORKING SESSION 1**

In this working session will be working with recursive code: understanding it first and implementing it second. The more you practice, the easier thinking recursively gets. Thus, this session is a lot about practice.

**Learning Objectives:**

1. **Trace** simple recursive functions, determining return values and being able to describe the task accomplished by the recursive function.
2. **Implement** the recursive version of an iterative algorithm
3. **Implement** simple recursive functions

|  |
| --- |
| **PART 1: TRACING RECURSIVE FUNCTIONS** |

For each of the following recursive functions:

1. Identify the line(s) defining the base case(s)
2. Identify the line(s) in charge of the recursive call
3. Determine the return value of the algorithm for a specific input data
4. Describe task accomplished by the recursive algorithm
5. **Implement it** in your computer (either using Java or C++, depending on the undergraduate programme you are pursuing) to verify whether your answer to c) and d) was correct.

|  |  |
| --- | --- |
| **Recursive function F1**  1. **function** F1(a,b)  2. **if** (b==0)  3. **return** 1  4. x=a\*F1(a,b-1)  5. **return** x | |
| Base case(s) line(s) | Line 2 |
| Recursive call line(s) | Line 4 |
| Return value for a=2, b=3 | 8 |
| Task performed by the recursive function | a^b |

|  |  |
| --- | --- |
| **Recursive function F2**  A: 1D array  N: number of elements of array A  1. **function** F2(A,N)  2. **if** (N==0)  3. **return** 0  4. **return** F2(A,N-1)+A[N-1] | |
| Base case(s) line(s) | Line 2 |
| Recursive call line(s) | Line 4 |
| Return value for A=[2,3,5] and N=3 | 10 |
| Task performed by the recursive function | Summing the array |

|  |  |
| --- | --- |
| **Recursive function F3**  A: 1D array  N: number of elements of array A  1. function F3(A,N):  2. if(N==1):  3. return A[0]  4. if(F3(A,N-1)<A[N-1]):  5. return F3(A,N-1)  6. return A[N-1] | |
| Base case(s) line(s) | Line 2 |
| Recursive call line(s) | Line 5 |
| Return value for A=[12,3,5] and N=3 | 3 |
| Task performed by the recursive function | It returns the mínimum value in the array |

|  |  |
| --- | --- |
| **Recursive function F4**  A: 1D array  low: lowest index  high: highest index  1. function F4(A,low,high):  2. if(high-low==1):  3. if(A[low]<A[high])  4. return A[low]  5. else  6. return A[high]  6. if (low==high)  7. return A[high]  8. mid=low+floor((high-low)/2)  9. a=F4(A,low,mid)  10. b=F4(A,mid+1,high)  11. if(a>b)  12. return b  13. return a | |
| Base case(s) line(s) | Line 2 to 7 |
| Recursive call line(s) | Line 10 and 11 |
| Return value for A=[12,3,5] and low=0 high=2 | 3 |
| Task performed by the recursive function | Finds the lowest value in array. |

|  |
| --- |
| **PART 2: FROM ITERATION TO RECURSION** |

Every iterative function can be transformed into a recursive function. Implement (using C++ or Java) the recursive version of the following iterative functions.

|  |  |
| --- | --- |
| **Iterative sum** | **Recursive sum** |
| a: integer number  b: integer number  function Sum(a,b)  result=0  for b<= i <=1  result=result+1  return a+result | Function(a,b)  If(B == 0){  Return b  }  Return a + function(A,b-1) |

|  |  |
| --- | --- |
| **Iterative subtraction** | **Recursive subtraction** |
| a: integer number  b: integer number  function Sub(a,b)  result=0  for b<= i <=1  result=result-1  return a+result | Function(a,b)  If(b == 0){  Return b  }  Return a – function(a,b-1) |

|  |  |
| --- | --- |
| **Iterative printing down** | **Recursive printing down** |
| a: integer number  function show(a)  for a<= i <=0  print(a) |  |

**HOMEWORK:**

|  |  |
| --- | --- |
| **Iterative printing up** | **Recursive printing up** |
| a: integer number  function show(a)  for 0<= i <=a  print(a) |  |

|  |
| --- |
| **PART 3: IMPLEMENTING RECURSIVE FUNCTIONS** |

Implement the recursive functions of the following numerical operations:

* Multiplication of two numbers, *a* and *b* (a\*b). Remember that the multiplication a\*b is the successive sum of the value a (*b* times).
* The integer division of two numbers, *a* and *b* (a/b). Remember that the division a/b consists in subtracting b units from a every time, until a condition is met (what is that condition?).

**HOMEWORK:** implement the modulus of two numbers (a%b).