Lebanese University
Faculty of Science
BS Computer Science
2nd Year – S3

I2204 - Imperative Programming

Dr Siba Haidar

Lebanese University
Faculty of Science
BS Computer Science
2nd Year – S3

Recursion

Chapter 1

Exercises

Testing a function

- In our programs
  - every time we will write a function
  - in order to test whether it gives us the result we want,
  - we must write a test function called the same but followed with Test word
- Example I
  - If the written function is callled myFunction
  - The test function must be called myFunctionTest

Full example

```
#include <stdio.h>
void myFunction(){
    printf("hello world!\n");
}

void myFunctionTest(){
    myFunction();
}

void main(){
    myFunctionTest();
    getchar();
}
```

Exercise : max2

- Write a C function "max2" that takes two integers as arguments and returns the value of the largest one.

- Write the following max2Test

```
void max2Test(){
      int a=1, b=3, max;
      max=max2(a,b);
      printf("the max(%d,%d)=%d\n",a,b,max
      max=max2(b,a);
      printf("the max(%d,%d)=%d\n",b,a,max);
}
```

- Write the main to call max2Test and verify your result

Exercise : max3

- Write a C function "max3" that takes three integers as arguments and returns the value of the largest one.

- Attention
  - max3 **MUST** call max2 instead of repeating the work
- write max3Test and call it from your main
  - Don't forget to delete the call for max2Test
- verify your result

Exercise : maxAll

- write a C function "maxAll" that takes an array of integers and returns the value of the largest one.

- attention
  - maxAll **MUST** call max2 instead of repeating the work
- write maxALLTest in which you initialize an array
  - int anArray[]={13,2,6,34,5,6,90,122,4,2,6,8,4,23,234};

- modify your main and test your function

Exercise: maxAll revisited

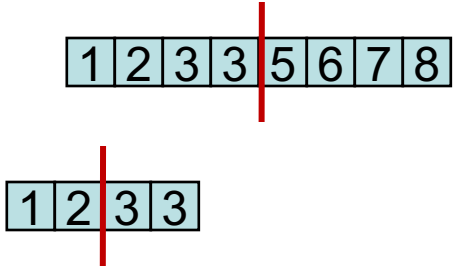
- write a **tail recursive** C function "maxAll" that takes an array of integers and returns the value of the largest one.
- write maxALLTest in which you initialize an array
  - int anArray[]={13,2,6,34,5,6,90,122,4,2,6,8,4,23,234};
- modify your main and test your function
- draw the memory state for
  - int anArray[]={13,2,6,34};

Exercise: merge

- write a C function ”merge" that takes two sorted arrays of integers and merges them into a third one.
- write mergeTest in which you declare 3 arrays and initialize the first two
- write main and test your function

Exercise: Binary Search

- write a tail recursive function
  - to apply binary search inside sorted arrays
  - find whether a given number n is inside the array
    - no → return -1
    - yes → return its first occurrence index
- example
  - find if 3 is there



Exercise: Fibonacci Function

- $$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

```
int F(int n){
    if (n == 0 || n == 1)
        return n;
    return F(n-1) + F(n-2);
}
```

Exercise: printArray

- Write a program in C to print the array elements using recursion.

Exercise: isPalindrome

- Write a program in C to check whether a given string is a palindrome or not.
  - Input a word to check for palindrome : mom
  - Expected Output :
    - The entered word is a palindrome.

Exercise: Quick Sort

- write in C the recursive function quicksort
- write quicksortTest

6 5 3 1 8 7 2 4

```
/* Double-Click To Select Code */

  function quicksort('array')
      if length('array') ≤ 1
         return 'array'  // an array of zero or one elements is already sorted
      select and remove a pivot value 'pivot' from 'array'
      create empty lists 'less' and 'greater'
      for each 'x' in 'array'
         if 'x' ≤ 'pivot' then append 'x' to 'less'
         else append 'x' to 'greater'
      return concatenate(quicksort('less'), 'pivot', quicksort('greater'))
// two recursive calls
```