

## CSE 1301 PRACTICAL SESSION 10

### Recursion (10 marks)

**Aim:** This practical session aims to give you practice at writing recursive programs.

**Coding style and documentation:** You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

#### **Preparation (to be completed before class)** **(2 marks)**

Write all the algorithms required for the questions below, and attempt the code for all the questions.

#### **PART 1: Power by iteration** **(1 marks)**

(a) Write an algorithm that reads in two integers, X and Y, and prints out the value of  $X^Y$  (X to the power of Y). Use iteration in your solution (and do not use the `pow()` function).

(b) Code, test and document your algorithm. Save your program as **Prac10-1.c**.

#### **PART 2: Power by recursion** **(2 marks)**

(a) Now write an algorithm that performs the same task on two integers, but now uses recursion, based on the following observations.

$$X^Y = \begin{cases} X^{Y/2} * X^{Y/2} & \text{when Y is even.} \\ X^{Y/2} * X^{Y/2} * X & \text{when Y is odd. Y/2 denotes integer division} \\ 1 & \text{when Y = 0} \\ X & \text{when Y = 1} \end{cases}$$

(b) Code, test and document your algorithm. Save your program as **Prac10-2.c**.

#### **PART 3: Greatest common divisor** **(2 marks)**

The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y. The gcd of x and y is defined recursively as follows:

If y is equal to 0, then

gcd(x, y) is x;

otherwise

gcd(x, y) is gcd(y, x%y) where % is the modulus operator.

Write a recursive function `gcd` that returns the greatest common divisor of x and y.

Code, test and document your function. Save your program as **Prac10-3.c**.

**PART 4: Palindromes** (“madam I’m adam”)**(3 marks)**

Write a program that inputs a word or phrase of unknown (but fixed maximal) length from the user, echoes it, and then prints it backward, exactly under the original. If the word or phrase is a palindrome, it will read the same forward as backward. Some examples of palindromes are:

“radar”

“able was I ere I saw elba” (attributed to Napoleon)

“pan a nap”

Write a recursive function that tests whether the string is a palindrome without explicitly reversing it. Use this function to test the input strings and display the message “Palindrome” instead of the reversed string if the string is one. Save your program as **Prac10-4.c**.

*Submission:*

Preparation: Show written preparation work to demonstrator at start of class for marking . If not done before class, it will not receive any marks, however you should still show it to your demonstrator during class to check your understanding.

PART 1, 2, 3, 4: To be marked by the demonstrator during this class. No late submission.

**Additional Component : Determinant Computation****(2 BONUS MARKS)**

Recall that the determinant of an  $n \times n$  matrix  $A$  is recursively defined as

$$|A| = \sum_{i=1}^n a_{i,j} \cdot C_{i,j}$$

where the  $C_{i,j}$  is the so-called co-factor

$$C_{i,j} = -1^{(i+j)} \cdot M_{i,j}$$

here  $M_{i,j}$  is a minor of  $A$ , i.e. the determinant of the sub-matrix obtained by deleting row  $i$  and column  $j$  from  $A$ . The determinant of a  $2 \times 2$  matrix is simply

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

This method is called co-factor development. According to this definition the determinant can be calculated by doing the following: go through the elements in one of the rows. For each element delete the row and column that intersect at this element and compute the determinant of this (smaller) matrix. Sum up (with strictly alternating signs) the determinants thus obtained for all elements in the row. This sum is the determinant of the whole matrix.

Write and code a recursive algorithm to compute the determinant of an  $n \times n$  matrix. Write a test program that reads such a matrix from the user and outputs its determinant.

*Note that this method would not be the method of choice in any real-world implementation. If you ever need to find efficient and advanced implementations for numerical problems, a good place to start your research is the the book “Numerical Recipes in C”.*