



File: SampleExercises-Selection-Loops.doc

Recommended exercises are pointed with blue arrows.

In the first steps, the student analyzing the problem and designing the program constructs algorithms using natural language, pseudocodes and flowcharts. Problem solving strategies, such as, IPO diagrams [IPO], condition-action decomposition [CAD], cursory sketches [CS], trace tables [TT], top down design [TDD], pseudocode [PSC], flowchart [FC] can help to understand and solve the problems. Later, after sometimes a python script (code, program) will be developed.

LOOPS

Ex-1. Loops. [M] Construct an algorithm to print a message (e.g., Hello There World) of your choice N times. For odd iterations the message should be "Hello There World, I Love you," while for even iterations the message should be "Hola Mundo, te quiero mas." Hint: should use **mod, rem, %** function or commands to determine if a number is divisible by 2 (even) or not (odd).



Ex-2. Loops. [H] Write and algorithm [CAD, CS, PSC, FC] to print the height of the tallest boy in a ten-people basketball team. Ask your program to assume the first boy is the tallest, then request the height of each boy one by one as it is compared to the previous one and the actual taller stored.

Ex-3. Loops. [M] Write an algorithm to print a table giving the purchase amount of gasoline from 1 liter to 25 liters, in steps of one. The price of gasoline is (currently) \$ 0.47 per liter.

Ex-XX. Loops. [M] Write an algorithm to print a table showing the area of circles with radii from 10-cm to 20-cm in steps of 0.5 cm.

Ex-XX. Loops. [M] Write an algorithm [PSC,FC] to compare covered area by 2.5 in and 5.0 in diameter circular cobblestones (adoquines) in a 10x10 feet square patio. Compute also how much area is covered by cobblestones on a 15x15 and 20x20 feet areas and implement a loop to accumulate the cobblestone areas. Report results in appropriate units.



Ex-4. Loops. Print the square roots of the first 25 odd positive integers.

Ex-5. Loops. [E] Write an algorithm to print the values of a squared numbers from 1 to 20.

Ex-6. Loops. [E] Write an algorithm to print the 9 times-table from 1 x 9 to 12 x 9.


Ex-8.a Loops. [M] Starting with one (1), write and algorithm to test for numbers less than 1000 and count how many of them are multiple of 3, 5, or 7 (If you were trying to develop a flowchart, you should read about 'modulo' flowgorithm operator, either mod or %; similar function or operator exist in most languages).

Ex-8.b. Loops. [M] Starting with one (1), write and algorithm to test for numbers less than 1000 and count how many of them are multiple of 3, 5, and 7 (If you were trying to develop a flowchart, you should read about 'modulo' flowgorithm operator, either mod or %; similar function or operator exist in most languages).

Ex-9. Loops. [E] Starting with [H+] concentration in a substance, write pseudocode to compute pH of several liquids as shown below and produce a table with "pH", "[H+] concentration", "type of substance" columns. Program requests from user the


substance type and the $[H^+]$ concentration until user types a sentinel (e.g., -1). To calculate the pH of an aqueous solution you need to know the concentration of the hydronium ion $[H^+]$ in moles per liter (molarity).

Type of Substance	$[H^+]$ Concentration moles/liter.	pH
battery acid	1	0
gastric acid	1.0E-01	1
lemon juice, vinegar	1.0E-02	2
orange juice, soda	1.0E-03	3
tomato juice, acid rain	1.0E-04	4
black coffee, bananas	1.0E-05	5
urine, milk	1.0E-06	6
pure water	1.0E-07	7
sea water, eggs	1.0E-08	8
baking soda	1.0E-09	9
Great Salt Lake, milk of magnesia	1.0E-10	10
ammonia solution	1.0E-11	11
soapy water	1.0E-12	12
bleach, oven cleaner	1.0E-13	13
liquid drain cleaner	1.0E-14	14

 **Ex-10. Loops. [E]** Write a flowchart or pseudocode to print the first 25 odd numbers which are also multiples of 7. Use the mod command or % operator in Flowgorithm and python.

Ex-11. Loops. [H] Write an algorithm to determine the factors of a positive number less than 100 entered by the user

Ex. 12. Loops. [M] Write an algorithm to play a number guessing game (User enters a guess, you print 'YES, you did it' or 'Higher' or 'Lower' in each attempt)

 **Ex-13. Loops. [H]** Write an algorithm for a program which figures out whether a given year is a leap year. In the Gregorian calendar three criteria must be taken into account to identify leap years:

1. The year can be evenly divided by 4, and
2. If the year can be evenly divided by 100, it is NOT a leap year, unless;
3. The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. Also 2016 and 2020 are leap years. User enters several years, the program reports if it is leap year or not until he decides entering -1 to stop.

SOLUTION. The table below is part of the problem decomposition, a year is chosen which wisely comply or not with one or more of the conditions:

Example Year	CONDITION			ACTION
	Div by 4	Div by 100	Div by 400	Leap?
2004	✓	☒		yes
2500	✓	✓	☒	no
2000	✓	✓	✓	yes
2003	☒			no

Ex-14. Loops. [M] Age Calculator. The user enters in the current year and then his/her birth year. Your program computes the user's age. Perform this task again for all her/his friends until user decides to stop.

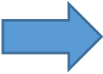
Ex-15. Loops. [M] Write a program which will repeatedly print the value of the positive variable **X**, decreasing it by 0.5 each time, as long as the **X** remains positive. Advice the user to enter initially a positive number.

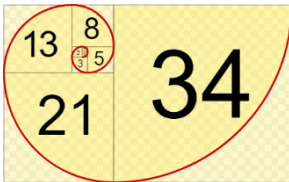
Ex-16. Loops. [E] Write an algorithm to print the square roots of the first 25 odd positive integers. Use the mod function

Ex-18. Loops. [M] Write pseudocode for a program which drives the user crazy by insisting they re-enter a particular input no matter what they enter. Be creative... You the programmer decides to stop the program after N times (once he/she is fully crazy).

Ex-19. Loops [H]. **Textito:** Program reads text represented as a sequence of characters. We want to calculate the number of occurrences of the letter **a** (or any character). We can assume that the text always has at least one character. Example, the entered text for the solution to the right was: "**Don Albertito Einstein**", a string with 21 characters which was entered (input) char-by-char. Then, program counted the number of "tees." Flowgorithm can't I/O the whole string in a single statement, thus you have to develop loops for input and output of strings char-by-char.

Ex-20. Loops. [H] If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000. **Hint:** Be careful not to add two times a number which is multiple of both 3 and 5.

 **Ex-21. Loops** [H, challenging] Each new term in the Fibonacci sequence is generated by adding the previous two terms. Starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...



By considering the terms in the Fibonacci sequence whose values do not exceed 1 million, find the sum of the even-valued terms.

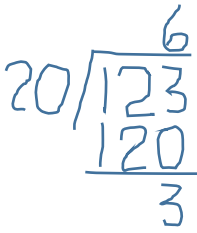
Ex 22. Loops (H) Write pseudocode to find the largest prime factors of 13195

Ex XX. Loops & Flags. Given the score of N students find if any got C. [$70 \leq \text{score} < 80$]

```

SOLUTION
SET FLAG=0, K=1
WHILE K<=N
    INPUT SCORE
    IF score>=70 AND score<80
        FLAG=1
    ENDIF
    K=K+1
ENDWHILE
IF FLAG==1
    PRINT "Some body got C"
ELSE
    PRINT "None got C"
ENDIF
    
```

Ex-23 Loops. [H] Write an algorithm to compute integer division. Integer Division (also called Long Division) is the process of division of two integers which produces a quotient and a remainder both integers. Subtract the divisor (n) from the dividend (d) successively until the result is smaller than the divisor ($n < d$). The quotient is the number of times the divisor is subtracted from the dividend (in the example below: 6).

Example	Algorithm
 <p>quotient</p> <p>remainder</p>	<pre> 123 20 ← 1 103 20 ← 2 83 20 ← 3 63 20 ← 4 43 20 ← 5 23 quotient is the number of times 20 ← 6 you subtract the denominator 3 ← remainder, stop when : n < d </pre>

Another algorithm:

Let **a** and **b** be positive integers. Then, there exist unique integers **q** [quotient] and **r** [remainder] such that: **a = bq + r** and **0 ≤ r < b**.

Ex-24 Loops. [H] GCD-Euclidean Algorithm. The Euclidean algorithm is one of the oldest numerical algorithms. It solves the problem of computing the greatest common divisor (gcd) of two positive integers (see problem description, example, and solution in document "Euclides Algorithm.pdf")

Ex-25 Loops. [M] Math Sequences. For the sequence 7, 10, 8, 11, 9, 12...write an algorithm to compute the first 10 numbers in this sequence.

[Part of the SOLUTION]: The pattern is increasing by 3, then decreasing by 2, so the next number in the series after 12 must be 10.

Ex-26. Loops. [M]. Sorting. Input three random integer values and print them in ascending order. E.g., test your code with A=3, B=2, C=1 and A=2, B=3, C=1.

% P#7.m

7.m

Sorts the list of three numbers from smallest to largest.

```

INPUT A
INPUT B
INPUT C
FOR k=1 to 2, Step 1
    if A>B
        X=A
        A=B
        B=X
    if B>C
        Y=B
        B=C
        C=Y

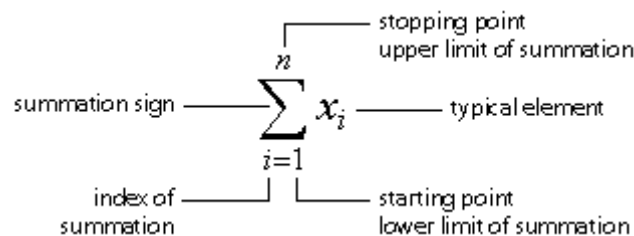
```

PRINT A, B, C

Ex-27. Loops [H]. Series. [Given pseudocode, write flowchart and python code] A geometric series is one where each successive term is produced by multiplying the previous term by a constant number (called the common ratio in this context). Example:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = \sum_{i=0}^{N=\infty} \frac{1}{2^i}$$

Where the summation sign means:



Write pseudocode to approximate the series with terms up to $N=10$. Once your code is bugs free change your program to ask N from the user.

Hint: you can use several approaches to solve this problem:

- (1) Using the modified expression $\sum_{i=0}^N \frac{1}{2^i}$, name numerator (n) equal to 1; name denominator (d) equal to 2^i ; define terms (t) as $t=n/d$; name the exponent i which will change as 0, 1, 2, 3, ..., N
- (2) We can also use $t_{i+1} = \left(\frac{1}{2}\right) t_i$ with $t_0 = 1$
- (3) We can use the expression for the denominator: $d=2*d$, with initial $d=1$

SOLUTION, pseudocode:

INPUT N

s=0

i=0

WHILE i<=N

n=1	# numerator (*)
d=2^i	# denominator
t=n/d	# a term of the series
s=s+t	# accumulation of each term
i=i+1	# counter or summatory index.

PRINT "The series is", s

(*) A better place is off the loop; no need to repeat each iteration $n=1$

Ex-28. Loops. Construct pseudocode and flowchart to determine if a given integer N is prime. What is the name of the algorithm you used to solve this problem? If you consult outside sources must provide references.

Ex-29. Loops. Construct pseudocode and flowchart to find the factors of N . Hint you can use the **mod** operator or function

Ex-30. Loops. Write pseudocode or flowchart to compute the factorial of N, i.e., $N! = 1 * 2 * 3 * \dots * (N-1) * N$

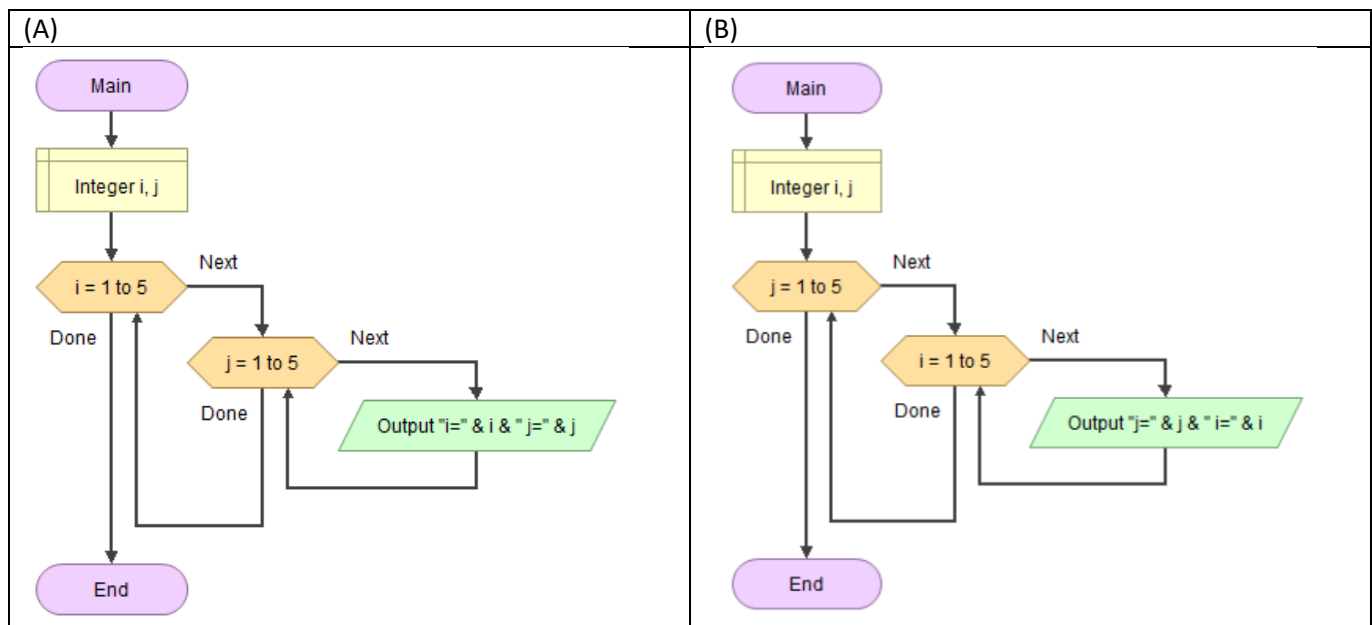
Ex-31. Loops. Nested Structures: Construct a trace table for the variable in the following code:

```

X=10
WHILE X>=1
    IF X<=5
        PRINT X
    ENDIF
    X=X-1
ENDWHILE

```

Ex-32. Loops. Nested Loops: Construct a trace table for the variables in the following flowchart:



Ex-2. Loops. [M]. Write an algorithm for the following guessing game. Ask for the radius of a circle, then keep asking guesses for the area of the circle and given clues to the user until the correct area is given.

Ex-7. Loops. [M] Write an algorithm for a program which keeps asking for a password made out of two integer numbers, advice if password is bigger or smaller, and to give the message "accepted" when the correct password is given. Program should report how many trials the user did.

Ex-17. Loops. [M] Write an algorithm to repeat a block of statements as long as the user indicates they want it to. This is an example of a sentinel (or flag) controlled loop.

LOOPS & ARRAYS (don't do the following sets; these problems are work in progress)

Ex-32. Input/Output the array $X = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$

Ex-33. Compute the average of the following set of values $A = [3, 4, 2, 6, 4, 3, 7, 8, 5, 4, 5, 6, 7]$;

Ex-34. Flip left to right (**fliplr**) the 1D array $ZAP = [1, 2, 3, \dots, 10]$; and store the results in the new array PAZ whose elements would be $PAZ = [10, 9, 7, \dots, 1]$.

Given A & B arrays (for Ex-28 through 31)

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

Ex-35. Transfer the diagonal elements of A to the diagonal elements of B

EX-36. Transfer the diagonal elements of A to the antidiagonal elements of B

EX-37. Multiply A diagonal elements by B antidiagonal elements and store them as the new B diagonal elements

EX-38. Multiply element-by-element the 3rd column of A times the 3rd row of B

FUNCTIONS

Ex-1-Functions. Write a program to print a table showing the area of circles with radii from 10-cm to 20-cm, steps of 1-cm. The area must be calculated using a user-defined function **circleArea**.

Ex-2. Compute the average of the following set of values $A=[0^2, 1^2, 2^2, 3^2, 4^2]$ The main program should I/O . Average is performed by calling the function **average**.

HARDER PROBLEMS

Ex-30. Sin(x) and Do/WHILE structure. Write pseudocode to compute the series $i\sin(x)$ for a given x value until error is less than 1×10^{-6}

$$i\sin(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{(2i+1)}}{(2i+1)!} = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Error is defined as

$$error = |true\ value - calculated\ value|$$



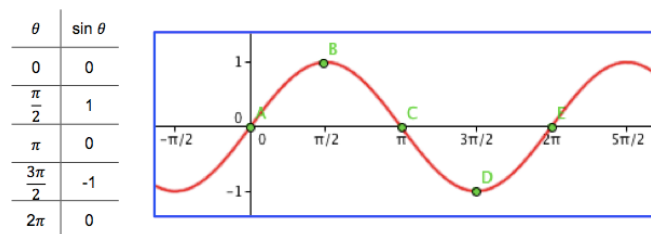
where calculated values is the value of the series as computed with the above equation and the true value is the value given by an efficient dwarf consulting the $\sin(x)$ tables for you living within any computer system, calculator, smart phone and providing you 24/7 with the precise result by free, i.e., the $\sin(x)$ library function.

The value 1×10^{-6} represents the allowed tolerance of the error.

The partial sum should be presented as a table with values of the angle x in radians.

Later we will extend the program objective to calculate $\sin(x)$ with values of the angle x in radians in the range of $[0, 2\pi]$ in steps of $\pi/4$.

RECALL:



A new loop structure is introduced which verifies the condition at the end. If the condition is true the loop continues. This structure called do/while executes the loop at least once.

```
do
    statements
while condition
```

Alternatively, we can write the above structure as:

```
while condition
    statements
    ...
    ...
    ...
end
```

Forcing the condition to be true initially by assuming smart values for the variable in the condition.

EX-31: Write pseudocode for the following game. The game is simple, the program thinks of a random number from 1 to 100, you (the user) make guesses and it tells you if you're too high or too low. When you find the number the program tells you the number and how many trials you did. This program teaches how to use the random number generator, basic input and output, loops and conditionals.

TOP DOWN DESIGN EXAMPLE

Ex-32. Write a program using top down design for the problem of computing the final score and final letter grade for the inge3016 MATLAB course of the whole class group with N students. Assume the final score is computed as:

$$\text{Final score} = \text{cp}(0.10) + \text{ex1}(0.15) + \text{ex2}(0.20) + \text{ex3}(0.25) + (\text{fal} + \text{qzs} + \text{att})(0.30)$$

where:

score (final score)

LG (letter grade)

cp (average score of computer projects, approximately 5-7 of them)

ex1, ex2, ex3 (score of first, second, ..., partial examinations)

fal (final examination score)

qzs (sum up all quizzes, each score is zero or one percent and qzs is an integer value)

abs (number of absences, and integer value)

att (bonus due to attendance, an integer value). Possibilities are:

If abs = 0 then assign att = 3 (perfect attendance)

If abs = 1 then assign att = 2

If $\text{abs} \geq 2$ then assign $\text{att} = 0$

The standard grading curve applies:

$\text{score} \geq 90\%$	letter grade is A
$80 \leq \text{score} < 90$	letter grade is B
$70 \leq \text{score} < 80$	letter grade is C
$60 \leq \text{score} < 70$	letter grade is D
$\text{score} < 50$	letter grade is F

Represent the weight of each score constituent with proper variables.