Computer Science 2020-21
Exercises

Teaching Assistant: ing. Gian Enrico Conti
10047232@polimi.it
Oct 2020

Ex1 loop: print even number

Ex1

Write a program that receives in input a positive integer value and outputs all its predecessors down to 0 if they are even:

(Hint: use "while" AND use "Modulo Operator" %)

Ex1 loop: print even numbers: Solution

Ex1

Write a program that receives in input a positive integer value and outputs all its predecessors down to 0 if they are even:

```
#ex 1
print("insert n:")
n = int(raw_input())
initial_n = n # save to exclude printing the same number we wrote in input

#skip first if not even: (we will decrement by 2 in loop)
remainder = n % 2
if remainder != 0:
    n-=1

while n>=0:
    remainder = n % 2
    if remainder == 0:
        if n != initial_n:
            print(n)
        n-=2
    else:
        n=-1
```

Ex2 loop: find max in input

Ex2

Write a program that receives in input 10 positive float values and outputs the number with heigher fractional part found, and the iteration where it was received.

Ex2 loop: find max in input

Ex2

Write a program that receives in input 10 positive float values and outputs the number with ~~heigher~~ fractional part found, and the iteration where it was received.

```
MAX_ITERATIONS = 10

i = 0
max = 0
max_fractional_part = 0.0
happenHere = 0
while i<MAX_ITERATIONS:
    n = float(raw_input())
    fractional_part = n - int(n)
    if fractional_part>max_fractional_part:
        max_fractional_part = fractional_part
        max = n
        happenHere = i
    i+=1

print("max: ", max, " at iteration: ", happenHere+1)
```

Ex3 loop: factorial

Ex3

Write a program that receives in input a positive integer value and outputs its factorial

(https://en.wikipedia.org/wiki/Factorial)

n	$n!$
0	1
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880

Ex3 loop: factorial: Solution

Ex3

Write a program that receives in input a positive integer value and outputs its factorial

```
print("n:")
N = int(raw_input())

fact = 1
while n>1:
    fact*=n
    n-=1

print("fact: ", fact)
```

Ex4 Common dividers

Ex4

Write a program that receives in input 2 positive integer values and outputs all their common dividers.

If no dividers, (for example n = 7, m = 3) outputs "No common dividers"

Example:
n = 120
n = 30

Ex4 Common dividers: Solution

Ex4

Write a program that receives in input 2 positive integer values and outputs all their common dividers.
If no dividers, (for example n = 7, m = 3) outputs "No common dividers"

```
print("n:")
n = int(raw_input())

print("m:")
m = int(raw_input())

count = 0
# stop at lower, so optimize loop:
if m>n:
    limit = n
else:
    limit = m

i = 2
while i<limit:
    if n % i == 0 and n  % i == 0 :
        print(i)
        count+=1
    i+=1

if count==0 :
    print("No common dividers")
```

```
for i in range(2, limit):
```

Ex5 Diving Contest

Ex5
During diving contest, points are assigned as follows:

5 competition judges assign a score each.

Final score is the double of the sum of all scores after removing higher and lower score.
Write a program that receives in input 5 positive integer values representing the scores and outputs the final score.

Example:

~~8.0~~, 7.5, 7.5, 7.5, ~~7.0~~ = 22.5 x 2.0 = 45.0

Ex5 Diving Contest: Solution

Ex5
During diving contest, points are assigned as follows:

5 competition judges assign a score each.
Final score is the double of the sum of all scores after removing higher and lower score.
Write a program that receives in input 5 positive integer values representing the scores and outputs the final score.

```
JUDGES = 5
max = 0
min = 10
sum = 0
i = 0

while i<JUDGES:
    vote = float(raw_input())
    sum+=vote
    if vote > max:
        max = vote
    if vote<min:
        min = vote
    i+=1

tot = (sum - max - min) * 2
print(tot)
```

Ex6 Pythagorean triplet

Ex6

Write a program that receives in input 2 positive integer values m and n, and outputs them if they form a Pythagorean triple.
Then program also outputs the other number (hypotenuse if we see them as a triangle)

Assume numbers lower than 30.

Example:

3, 4 output: 3, 4, 5
6, 8 output: 6, 8 10

Ex6 Pythagorean triplet: Solution

Ex6

Write a code that reads TWO positive numbers N, M in input and find if they generate a Pythagorean triple printing also the other number (hypotenuse if we see them as a triangle)

```
n = int(raw_input())
m = int(raw_input())

i = 2
limit = 30 * 30
while i<limit:
    if n * n + m * m == i * i:
        print(m, n, i)
    i+=1
```

Ex7 longest growing sequence

Ex7

Write a program that asks the user that acquires an arbitrary sequence of ~~positive~~ integers, terminated when s/he inputs 0, computing and outputting at the end of the sequence, le longest growing sequence length.

Example:

3 8 4 5 1 17 0
max len = 2

19 18 14 9 6 4 3 0
max len = 1

1 3 6 8 0 1 12 0
max len = 4

Ex7 longest growing sequence

Ex7

Write a program that asks the user that acquires an arbitrary sequence of positive integers, terminated when s/he inputs 0, computing and outputting at the end of the sequence, le longest growing sequence length.

```
oldValue = 0
maxGrowingSequenceLen = 0
currGrowingSequenceLen = 0
n = 1

while n!=0:
    n = float(raw_input())
    if n>oldValue:
        currGrowingSequenceLen+=1
        oldValue = n
    else:
        oldValue = 0
        currGrowingSequenceLen = 0

    if currGrowingSequenceLen>maxGrowingSequenceLen:
        maxGrowingSequenceLen = currGrowingSequenceLen

print(maxGrowingSequenceLen)
```

Ex8 parity

Ex7

In computer science there is a technique to detect errors, called parity. By definition we compute parity only on number less than 128.

There are two types of parity, Even and Odd, that must be set before transmission.

Parity bit is a bit added during transmission. Let's call it PB.

First step is to compute the number of "1" bits.

- if parity chosen is Even, total number of "1" bits **including** PB must be even.
- if parity chosen is Odd, total number of "1" bits **including** PB must be odd.

Example:
if n is 77, (1001101 in binary) "1" bits are 4, so if parity is Even, PB will be 0 (4 total "1")

Write a program that receives in input a positive integer n, and outputs Parity Bit in case of "even" parity

Hint: divide number by 2 to get single bits as remainder.

2nd part: modify program to compute Parity Bit in case of "odd" parity.

Ex8 parity: solution

Ex7

Write a program that receives in input a positive integer n, and outputs Parity Bit in case of "even" parity

2nd part: modify program to compute Parity Bit in case of "odd" parity.

```
#ex 7 parity

ODD_PARITY = 1
EVEN_PARITY = 0      BASE = 2

parity = EVEN_PARITY #let's assume even

n = int(raw_input())
numberOf1s = 0

while n>0:
    remainder = n % 2
    if remainder == 1:
        numberOf1s +=1
    n = n /2

parityBit = (numberOf1s + parity) % 2
print(parityBit)
```

Ex9 temperature search

Ex8

We want to process temperatures of some physical room in our plant.
Room are mapped using ~~an~~ x,y coordinates relative to center of the plant (0,0)

Write a program that first asks the user coordinates of her/his place, then asks how many triplets composed by x, y and temperature s/he will input, and then it acquires them all, computing and outputting at the end of the sequence, the temperature, x and y of the **nearest** place of received input values and TRUE if is it is also the hottest.

Ex9 temperature search

Ex8

Write a program that first asks the user coordinates of her/his place, then asks how many triplets composed by x, y and temperature s/he will input, and then it acquires them all, computing and outputting at the end of the sequence, the temperature, x and y of the **nearest** place of received input values and TRUE if is it is also the hottest.

```
INVALID_COORD = -1.0
SQUARE_BIG_DISTANCE = 10000000
userX = float(raw_input())
userY = float(raw_input())
numberOfPlaces = int(raw_input())
maxTemperature = -273.0  # every temperature will be heigher.
minSquareDistance = SQUARE_BIG_DISTANCE
nearerX = INVALID_COORD
nearerY = INVALID_COORD
nearerTemperature = INVALID_COORD
indexOfNearest = 0
indexOfHottest = 0
n=0
while n<numberOfPlaces:
   x = float(raw_input())
    y = float(raw_input())
   temperature = float(raw_input())
    deltaX = (userX - x)
    deltaY = (userY - y)
    squareOfDistance = deltaX * deltaX + deltaY * deltaY

    if squareOfDistance<minSquareDistance:
        nearerX = x
        nearerY = y
        nearerTemperature = temperature
        indexOfNearest = n
        minSquareDistance = squareOfDistance
    if temperature>maxTemperature:
        maxTemperature = temperature
        indexOfHottest = n
    n+=1

print(nearerX, nearerY, nearerTemperature)
if indexOfNearest == indexOfHottest:
    print(True)
```