German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Amr Desouky

January 21, 2016

CSEN 102: Introduction to Computer Science
Winter term 2015-2016
Final Exam

Bar Code

Instructions: Read carefully before proceeding.

(a) Please tick your major

	Major
	Engineering
	BI

(b) Duration of the exam: 3 hours (180 minutes).

(c) (Non-programmable) Calculators are allowed.

(d) No books or other aids are permitted for this test.

(e) This exam booklet contains 16 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.

(f) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.

(g) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below ;-)

Exercise	1	2	3	4	5	6	7	$\sum$
Possible Marks	10	10	12	6	14	11	17	80
Final Marks								

Exercise 1 General arithmetics (6+4=10 Marks)
Number conversions

In this exercise you have to convert numbers from one base to another. A number n of a base b other than 10 will be given on this page as

$$n_b$$

For decimal numbers, the subscript base can be omitted (*i. e.*, we write n instead of "n_{10}"). For bases b that are greater than 10, please use uppercase Latin letters (A, B, ...) as digits beyond 9, just as it is done with hexadecimal numbers.

(a) Convert the following numbers into binary. Show your workout. (6 Marks)

1. 37_8

   Solution:
   $37_8 = 11111_2$
2. AF_{16}

   Solution:
   $AF_{16} = 10101111_2$
3. 5321_5

   Solution:
   The conversion is not applicable since the given number is not correct.

(b) Convert the following and show your workout. (4 Marks)

- CB_{13} into hexadecimal base (base 16) number

  Solution:
  $CB_{13} = A7_{16}$
- 223_4 into octal base (base 8) number

  Solution:
  $223_4 = 53_8$

Exercise 2 Binary representations (6+4=10 Marks)
Normalized scientific binary floating point

Recall the encoding schema for a normalized scientific binary floating point from Lecture 8:

Sign of mantissa 1 bit	Mantissa x bits	Sign of exponent 1 bit	Exponent y bits

(a) We would like to translate the number 63.125_{10} into this schema in an accurate way. Your task is first to decide about the least values of x that corresponds to the size of the mantissa and y that corresponds to the size of the exponent (without the sign bits). Show your workout. (6 Marks)

Solution:

1. The number $63.125_{10} = 111111.001_2$
2. The number in normalized scientific notation: $.111111001 * 2^{+6}$
3. $x = 9$ bits and $y = 3bits$

(b) What is the decimal value of the corresponding normalized scientific binary floating point? (4 Marks)

1	1110	1	011

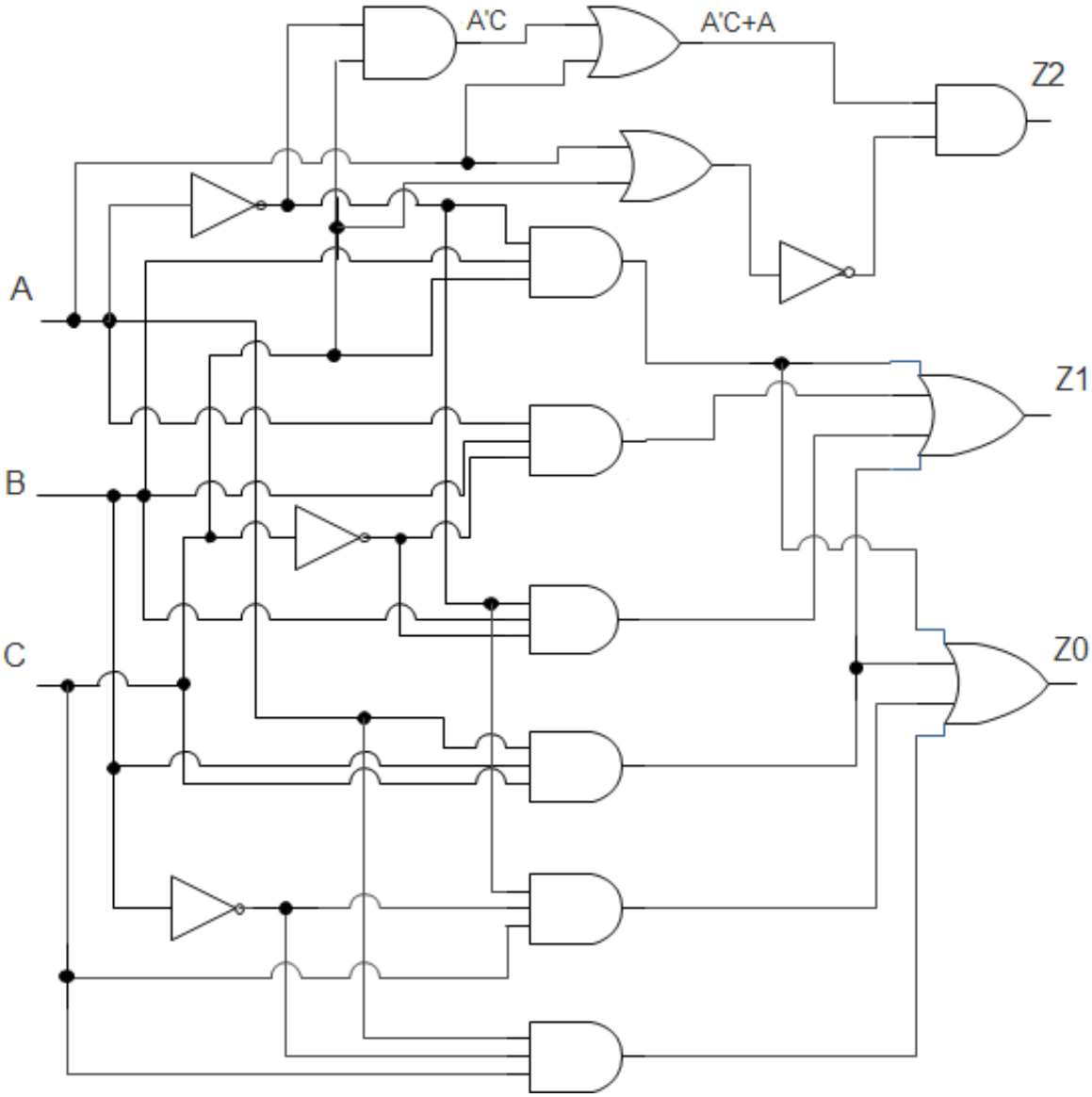
Solution:

1. The number in normalized scientific notation: $-0.111 * 2^{-3}$
2. In decimal: -0.109375

Exercise 3 Boolean functions (6+4+2=12 Marks)
Circuit improvement

For this exercise, we can use AND and OR gates with more than 2 inputs. For example, we can use the following AND gate:

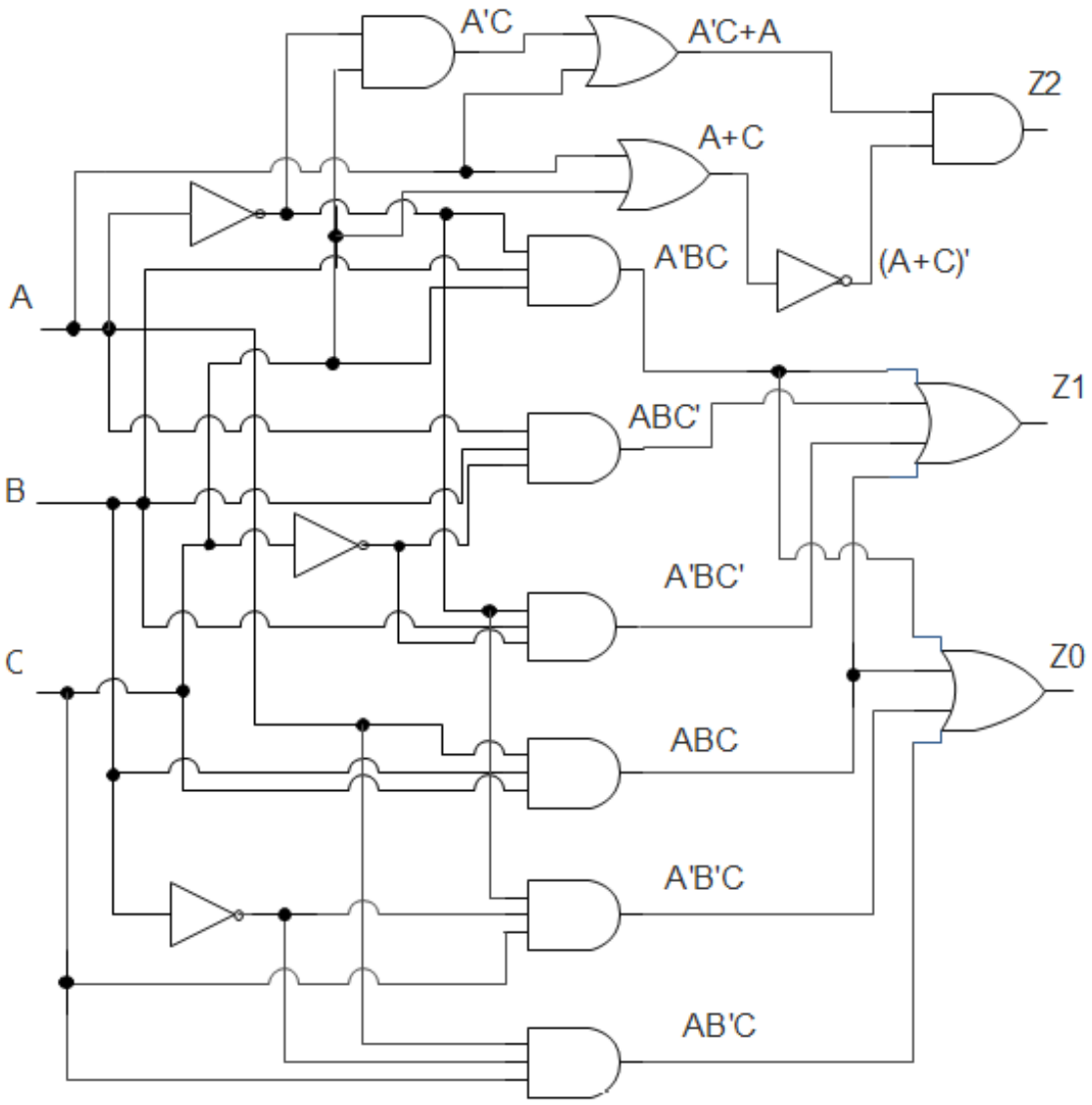
Given the following circuit, where A, B and C are the input variables and $Z2$, $Z1$ and $Z0$ are the output variables (where $Z2$ is the left-most bit in the output):



(a) Extract the Boolean expressions for the output bits from the circuit. Justify your answer by writing the corresponding expression on the output of every gate in the circuit (as shown on the circuit). (6 Marks)

Solution:

$Z2 = (A'C + A) * (A + C)'$
$Z1 = ABC + ABC' + A'BC + A'BC'$
$Z0 = AB'C + A'B'C + ABC + A'BC$



(b) Simplify the expressions using the aximos of the Boolean Algebra. Please mention the applied rules.

$x + 0 = x$	$x * 1 = x$	
$x + 1 = 1$	$x * 0 = 0$	
$x + x = x$	$x * x = x$	
$x + x' = 1$	$x * x' = 0$	
$(x')' = x$		
$x + y = y + x$	$xy = yx$	Commutativity
$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$	Associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	Distributivity
$(x + y)' = x'y'$	$(xy)' = x' + y'$	DeMorgan's Law

Note that the simplified expressions would not consist of any Boolean operations. (4 Marks)

Solution:

$$
\begin{aligned}
Z2 = & \quad (A'C + A) * (A + C)' & \\
= & \quad (A'C + A) * (A'C') & \text{(DeMorgan's Law)} \\
= & \quad A'CA'C' + AA'C' & \text{(Associativity)} \\
= & \quad 0 + 0 & \text{(x * x' = 0)} \\
= & \quad 0 & \text{(x + 0 = x)}
\end{aligned}
$$

$$
\begin{aligned}
Z1 = & \quad ABC + ABC' + A'BC + A'BC' & \\
= & \quad AB(C + C') + A'B(C + C') & \text{(Associativity)} \\
= & \quad AB + A'B & \text{(x + x' = 1)} \\
= & \quad B(A + A') & \text{(Associativity)} \\
= & \quad B & \text{(x + x' = 1)}
\end{aligned}
$$

$$
\begin{aligned}
Z0 = & \quad AB'C + A'B'C + ABC + A'BC & \\
= & \quad B'C(A + A') + BC(A + A') & \text{(Associativity)} \\
= & \quad B'C + BC & \text{(x + x' = 1)} \\
= & \quad C(B + B') & \text{(Associativity)} \\
= & \quad C & \text{(x + x' = 1)}
\end{aligned}
$$

(c) What is the functionality of the circuit? (2 Mark)

Solution:

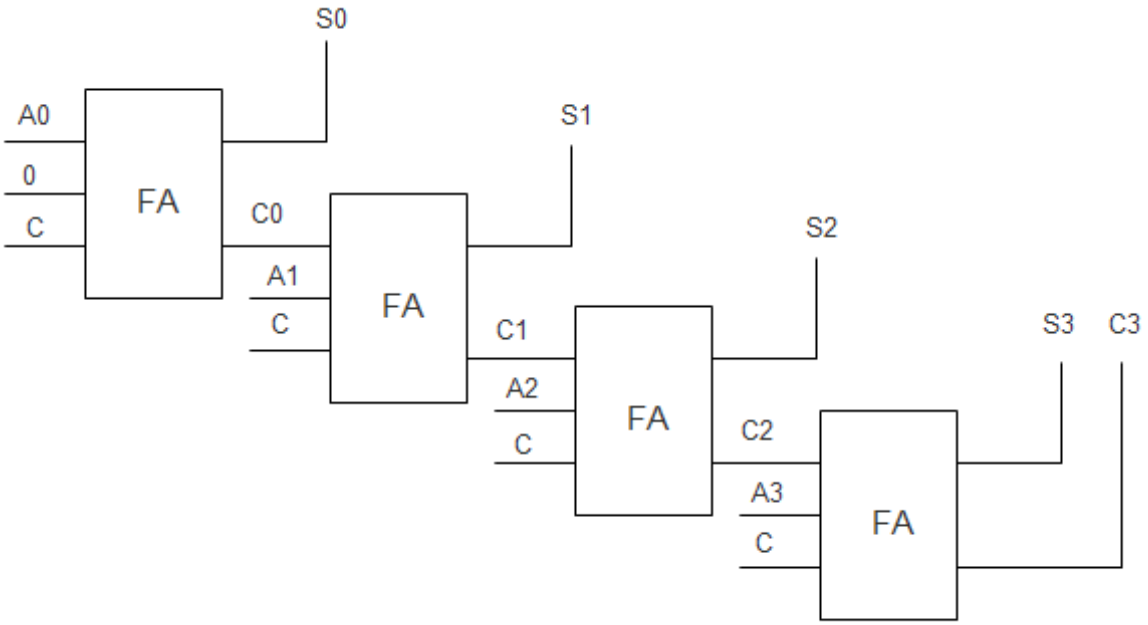
The circuit gets the absolute value of an input number given in sign/magnitude notation.

Exercise 4 Boolean Circuits (6 Marks)
Full-Adders

Given only full-adders, design a single circuit that performs the following operation on a 4-bit number A, where A is represented in two's complement. **Note** that A can be positive or negative.

$$\mathrm{F(A, C)} = \begin{cases} A & \text{if } C = 0 \\ A - 1 & \text{if } C = 1 \end{cases}$$

Solution:



Exercise 5 Boolean Circuits (2+2+4+6+(4 Bonus)=14 Marks)

n-to-m-Line Decoders

An n-to-m-line decoder is a circuit having n inputs and m outputs, where $m = 2^n$. Such that for any input x, the output is 2^x and exactly one of the output variables will be active, i.e. equal to one.

For example, a 1-to-2-line decoder has the following truth table:

x	O_1	O_0
0	0	1
1	1	0

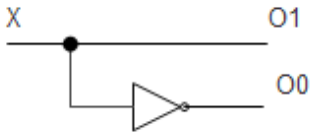
(a) Using the sum-of-products method, write Boolean expressions for a 1-to-2-line decoder. (2 Marks)

Solution:

$O1 = X$
$O0 = X'$

(b) Draw the circuit for a 1-to-2-line decoder. (2 Marks)

Solution:

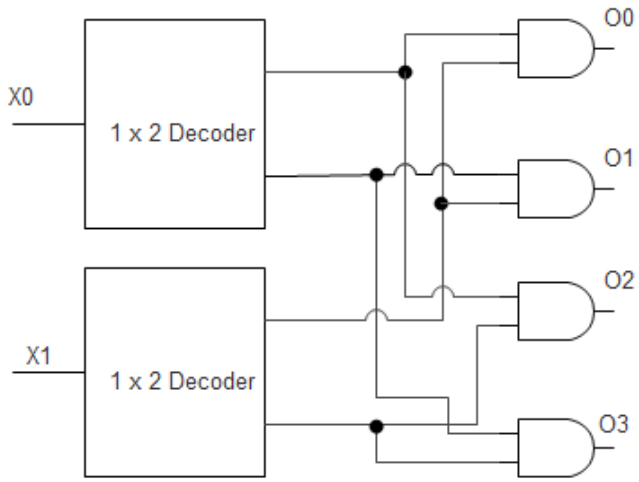


(c) Draw the truth table of 2-to-4 line decoder. (4 Marks)

(d) Assume that you have already manufactured 1-to-2-line decoders.

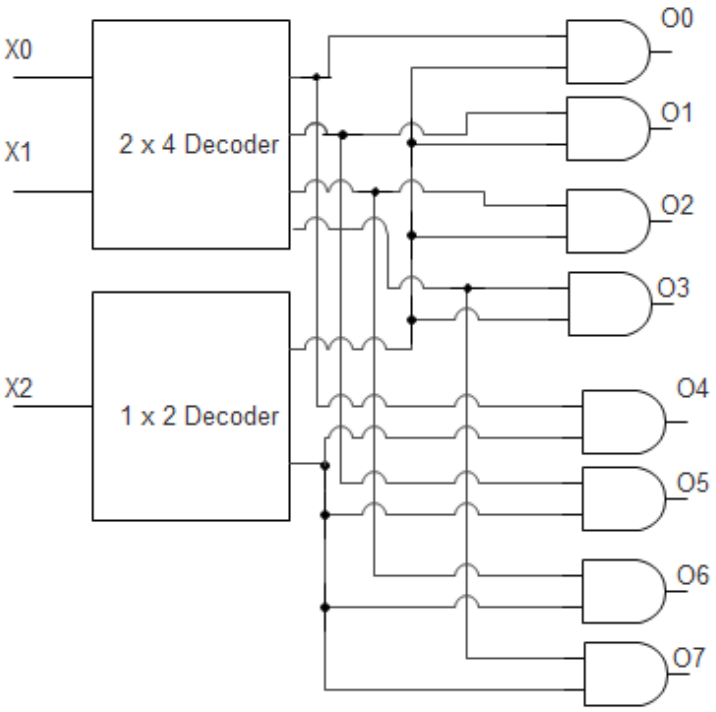
Design a circuit that uses only 1-to-2-line decoders and AND-gates to simulate the functionally of a 2-to-4 line decoder. (6 Marks)

Solution:



(e) Having manufactured 1-to-2-line decoders and 2-to-4-line decoders, draw a circuit to make a 3-to-8-line decoder. (4 Marks (Bonus))

Solution:



Exercise 6 Iterative operations (8+3=11 Marks)

Mysterious Number 6174

Why is the number 6174 so interesting? As defined by Wikipedia

1) Take any four-digit number, having at least two different digits. (Leading zeros are allowed.)
2) Arrange the digits in ascending and then in descending order to get two four-digit numbers, adding leading zeros if necessary.
3) Subtract the smaller number from the bigger number.
4) Go back to step 2).

The above process, known as Kaprekar's routine, will always reach 6174 in at most 7 iterations. Once 6174 is reached, the process will continue yielding it. Thus you should stop.

(a) Write a program which runs the Kaprekar's routine for a given four-digit **number** (see definition above) **printing out each step of the routine**.

For example:

- For the input `2607`, the program should display:

```
7620 - 0267 = 7353
7533 - 3357 = 4176
7641 - 1467 = 6174
Iterations: 3.
```

- For the input `1211`, the program should display:

```
2111 - 1112 = 0999
9990 - 0999 = 8991
9981 - 1899 = 8082
8820 - 0288 = 8532
8532 - 2358 = 6174
Iterations: 5
```

Hint: you can use the sort() function that sorts a list in an ascending order.

For example:

```
A = [5,2,4,9,1,6]
A = A.sort()
print(A)
```

will print:

```
[1, 2, 4, 5, 6, 9]
```

Solution:

```
x = eval(input())
i = 0
A = [0]*4
while(x != 6174):
  A[3] = x%10
  A[2] = (x//10) % 10
  A[1] = (x//100) % 10
  A[0] = (x//1000) % 10
  A = A.sort()
```

```
    num1 = A[0]*1000 + A[1]*100 + A[2]*10 + A[3]
    num2 = A[3]*1000 + A[2]*100 + A[1]*10 + A[0]
    x = num2 - num1
    print(num2,"-",num1,"=",x)
    i +=1
print("Iterations",i)
```

(b) What is the order magnitude of the program for an input number consisting of four digits where at least two digits are different? Justify your answer.

Solution:

The order of magnitude is $O(1)$ since for the worst case, the maximum number of iterations that could be executed is 7 iterations. Thus, it is constant.

Exercise 7 Algorithms and efficiency (6+3+6+2=17 Marks)

Mysterious

Given the following algorithm:

```
x = input()
even = ""
odd = ""
i = 0
j = 1
while i < len(x):
  even = even + x[i]
  i += 2
while j < len(x):
  odd = odd + x[j]
  j += 2
evenS = 0
i = 0
while i < len(even):
  if (i%2 == 0):
    evenS += int(even[i])
  else:
    evenS -= int(even[i])
  i += 1
oddS = 0
i = 0
while i < len(odd):
  if (i%2 == 0):
    oddS += int(odd[i])
  else:
    oddS -= int(odd[i])
  i += 1
sum = 2*oddS + evenS
if(sum%5 == 0 and x[len(x) - 1] =='0'):
  print("The input has the mysterious property")
else:
  print("The input does not have the mysterious property")
```

(a) Trace your algorithm for `x = "110010"` and state what will be displayed. (6 Marks)

Solution:

```
even = "101"
odd = "100"
evenS = 1+1 = 2 - 0 = 2
oddS = 1
sum = 2 + 2 = 4
"Doesn't have the mysterious property"
```

(b) What does the algorithm do for any binary sequence as input? Find the property for the input variable `x`. (3 Marks)

Solution:

It checks whether the given input in binary is divisible by 10.

(c) Give the total number of executed operation and the order of magnitude for the running time of the program. Carefully justify your answer. (6 + 2 Marks)

```
x = input()   ----------> 1 operation executed once
even = ""
odd = ""
i = 0
j = 1
while i < len(x):
  even = even + x[i]
  i += 2
while j < len(x):
  odd = odd + x[j]
  j += 2
evenS = 0
i = 0
while i < len(even):
  if (i%2 == 0):
    evenS += int(even[i])
  else:
    evenS -= int(even[i])
  i += 1
oddS = 0
i = 0
while i < len(odd):
  if (i%2 == 0):
    oddS += int(odd[i])
  else:
    oddS -= int(odd[i])
  i += 1
sum = 2*oddS + evenS
if(sum%5 == 0 and x[len(x) - 1] =='0'):
  print("The input has the mysterious property")
else:
  print("The input does not have the mysterious property")
```

Total number of executed instructions in terms of n, where n is the length of the input String:

Order of magnitude:

Solution:

Number of executed instructions: 7*ceil(n/2) + 7*floor(n/2) + 17

Order of magnitude: 0(n)