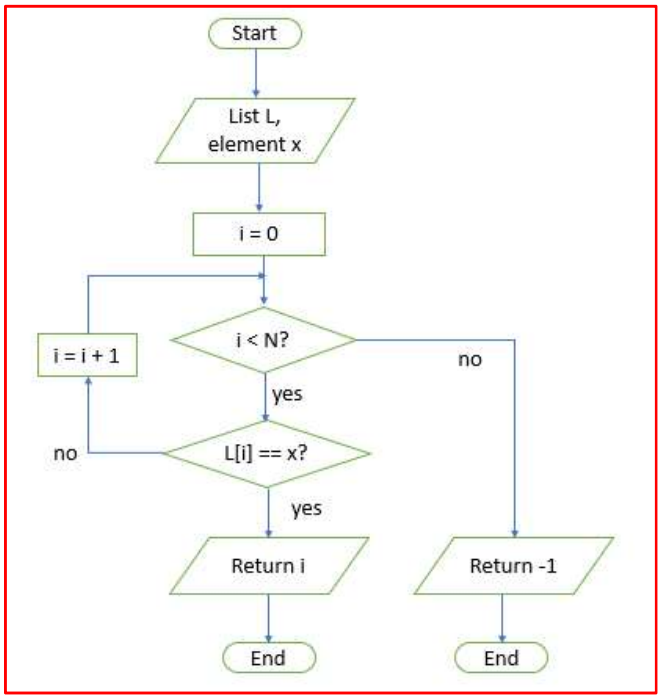
# 1. CSE101 – Introduction to Computers

## End-term Review

2. A. Match the following pairs.

| | |
|---|---|
| Computation | A description of solving the problem which includes a precise statement of the problem (the input), the desired solution (the output), and the order in which the steps will be executed |
| Algorithm | A problem might not be solvable by computation because it is ambiguous, it requires too many steps to complete, or it is mathematically impossible |
| Limitations | sequence of simple, well-defined steps carried out to solve a problem |

B. Given a list of N elements and element x, create a flowchart of a linear search algorithm which will return the index of x if element x is present in the list or return -1 otherwise.




C. Give an example of problem which is computationally huge such that it is unsolvable by a computer.

If a computer tries to analyze every possible sequence of moves in response to this opening in a game of chess, it will have to consider over 1043 different games.
Computer solving one trillion combinations per second will compute the perfect game of chess if we are patient enough to wait 1021 years, so it is only unsolvable in a practical sense.

3. A. Given x = 10, what will be the result of evaluating the following expressions?

```
((x / 2) ** 2) / 2 = 12.5
(x / 2) ** 2 / 2 =   12.5
x / 2 ** 2 / 2 =     1.25
```

B. Suppose the following strings are defined in an interactive session:

```
>>> s = "baseball"
>>> t = "The Korea Baseball Championship, is the highest level league of baseball in South Korea."
```

What will Python print as the value of the following expressions?

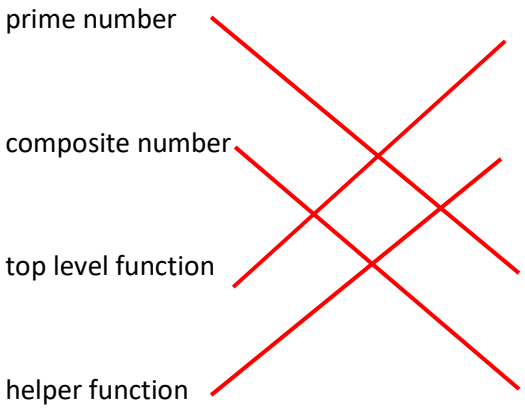
```
(s + '!') * 2 = baseball!baseball!
s in t = True
t.count('o') = 5
```

C. Define a function named `pmt` that will compute and return the amount of a monthly payment on a loan. The three parameters of the function should be amt, the initial loan amount; rate, the annual interest rate and yrs, the number of years before the loan is paid off. The algorithm for computing the payment is as follows. First, calculate a value r using the formula `r = rate/100/12`. Then calculate a value `p = 12 x yrs`. The formula for the payment is then:

$$\text{payment} = \frac{r \times \text{amt}}{1-(1+r)^{p}}$$

```
def pmt(amt, rate, yrs):
    r = rate/100/12
    p = 12 * yrs
    payment = (r * amt)/(1 - (1 + r)**p)
    return payment
```

4. A. Match the following pairs:




A function that implements a complete solution to a problem

A function designed to solve a small part of a larger problem

An integer that is not evenly divisible by any numbers except 1 and itself

An integer that can be expressed as the product to two or more other integers

B. Suppose a list is defined with this statement:

```
>>> gas = ['He', 'Ne', 'Ar', 'Kr', 'Xe', 'Rn']
```

What are the values of the following Python expressions?

```
'Ne' in gas = True
'Fe' in gas = False
gas.index('Ne') = 1
gas.index('Xe') = 4
```

C. Suppose we define two lists of numbers as follows:

```
>>> a = [1,1,2,3,5,8]
>>> b = [13, 21, 34]
```

Explain what are the values of the following Python expressions?

```
a[0] + b[0] = 14
a + b = [1,1,2,3,5,8,13,21,34]
```

5. A. Assume a list is defined with this statement:

```
>>> heavy = ['U', 'Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf']
```

Explain how the list would be sorted by a call to isort. Here is the first line to get you started:

```
>>> isort(heavy)
['U'] ['Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf']
```

**['Np', 'U'] ['Pu', 'Am', 'Cm', 'Bk', 'Cf']**

**['Np', 'Pu', 'U'] ['Am', 'Cm', 'Bk', 'Cf']**

**['Am', 'Np', 'Pu', 'U'] ['Bk', 'Cf']**

**['Am', 'Cm', 'Np', 'Pu', 'U'] ['Bk', 'Cf']**

**['Am', 'Bk', 'Cm', 'Np', 'Pu', 'U'] ['Cf']**

**['Am', 'Bk', 'Cm', 'Np', 'Pu', 'U'] ['Cf']**

B. Define a Python function named `gcd` that will compute the greatest common divisor of two integers a and b using Euclid's algorithm, another early algorithm that is over two thousand years old. The pseudocode of this algorithm is given below. In modern terminology, the algorithm uses a while loop that terminates when a = b. In the body of the loop, compare a to b and subtract the smaller value from the larger one. When the loop terminates, return a as the result of the call.

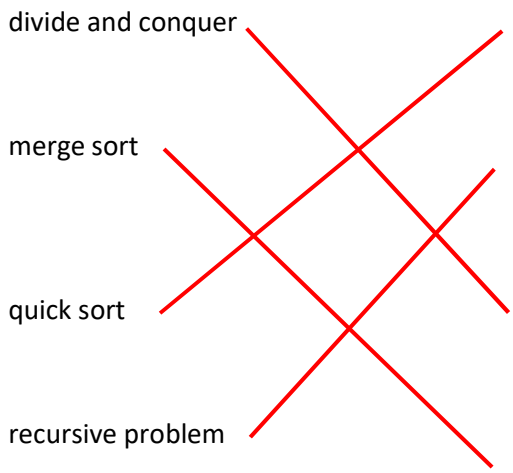
Pseudocode of Euclid's GCD algorithm:
(:= represents assignment)

```
function gcd(a, b)
    while a ≠ b
        if a > b
            a := a - b;
        else
            b := b - a;
    return a;
```

Corresponding Python code:

```
def gcd (a, b):
  while a != b:
    if a > b:
      a = a – b
    else:
      b = b – a
  return a
```

6. A. Match the following pairs:




An algorithm that sorts a list through a top-down application of the divide an conquer strategy

A problem that can be broken into one or more subproblems that are each smaller instances of the main problem

A problem-solving strategy that breaks a problem into smaller pieces and addresses each subproblem separately

An algorithm that sorts a list by combining small groups into larger groups, using a bottom-up application of the divide and conquer strategy

B. Write both, an iterative function and a recursive function that returns the n-th number in a Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, …..

Iterative solution:

```
def fib(n):
    fibl = [0,1]
    for i in range (2, n):
        fibl.append(fibl[i-2] + fibl[i-1])
    return fibl
```

Recursive solution:

```
# Returns the n-th Fibonacci number
def fib(n):
    if n == 0 or n == 1:  # two base cases
        return 1
    return fib(n - 1) + fib(n - 2)  # two recursive calls
```

7. A. Write an assignment statement that create dictionary named continents for the seven continents on the earth which are: `Asia`, `Africa`, `North America`, `South America`, `Europe`, `Oceania`, `Antarctica`. In this dictionary, the first two characters of the continent name is the key and the name of the continent is value.

```
continents = {"As":"Asia", "Af":"Africa", "No":"North America",
"So":"South America", "Eu":"Europe", "Oc":"Oceania", "An":"Antartica"}
```

B. Write a function, *acronym*, that creates an acronym from the first letter of each long word in a list, where a long word is any word with more than three letters.

```
>>> acronym('operating system')
'OS'
>>> acronym('association for computing machinery')
'ACM'
```

```
def acronym(phrase):
    result = ''
    words = phrase.split()
    for w in words:
        if len(w) > 3:
            result += w.upper()[0]
    return result
```

8. (Pseudo-Random Numbers) Study the random number generator code here:

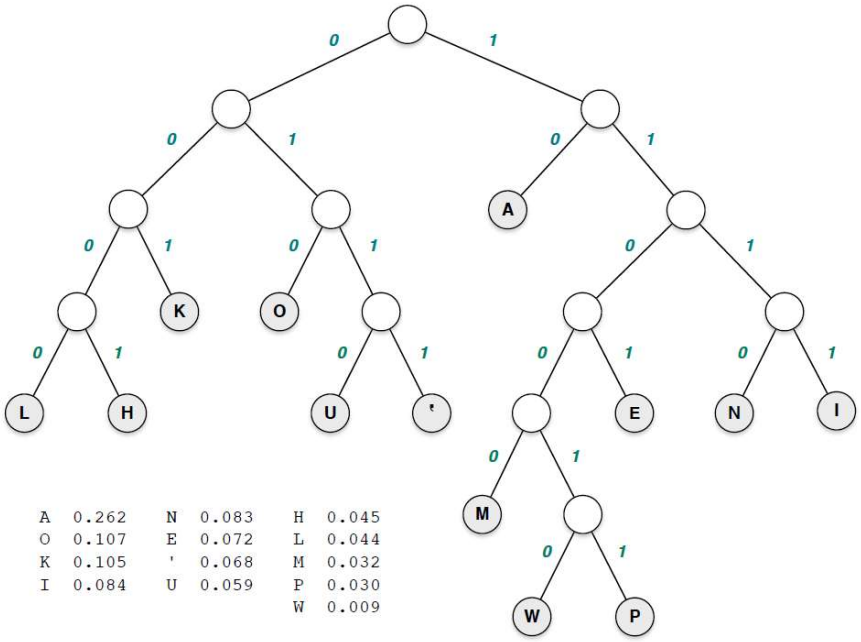
```
a = 4
c = 11
m = 23
x = 3 % m

def rand():
    global x
    x = (a * x + c) % m
    return x

for i in range(10):
    print (str(rand()) + " ", end="")
```

What 10 values will this code generate?

____ 0 11 9 1 15 2 19 18 14 21 _____

_____

9. (Huffman Codes)




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | 0.262 | N | 0.083 | H | 0.045 | | |
| O | 0.107 | E | 0.072 | L | 0.044 | | |
| K | 0.105 | ' | 0.068 | M | 0.032 | | |
| I | 0.084 | U | 0.059 | P | 0.030 | | |
| | | | | W | 0.009 | | |

a) Above is the Huffman tree for the Hawaiian alphabet. Using the tree and the labeled arcs, write the bit code for each letter in the table below.

| Letter | Code | Letter | Code |
|---|---|---|---|
| ‘ | 0111 | l | 0000 |
| a | 10 | m | 11000 |
| e | 1101 | n | 1110 |
| h | 0001 | o | 010 |
| i | 1111 | p | 110011 |
| k | 001 | w | 110010 |
| u | 0110 | | |

b) Now encode the Hawaiian words below

‘A‘olepilikia ___0111 10 0111 010 0000 1101 110011 1111 0000 1111 001 1111 10__

Hoaloha ___0001 010 10 0000 010 0001 10_____

Mo‘opuna ___11000 010 0111 010 110011 0110 1110 10_____

10. (Classes and OOP)

Create a class called `Worker`. *Worker* holds information on a factory worker in a company. The information includes the worker's full name, hourly rate, hours in a standard week and hours in an extended week.

A worker earns their normal hourly rate for the number of hours in a standard week. If they work more hours, for the extra hours, they earn 1.5 times their hourly rate. Finally, if they work beyond the number of hours in the extended week, any hours over that number are paid at 2 times the hourly rate.

The class must have an __init__ method to build the object given the worker's name, hourly rate, standard hours, and extended hours.

You must also write a `calculatePay()` method that takes the number of hours worked that week and returns the amount of pay in US dollars.

Example: If Joe Cool has a standard work week of 40 hours, an extended week of 50 hours, and wage of 18.50 per hour, his pay for 55 hours would be:

40 * 18.50 + 10 * 18.50 * 1.5 + 5 * 18.50 * 2 = 1202.50

So creating a *Worker* object to compute this would look like:

```
w = Worker ("Joe Cool", 40, 50, 18.50)
print (w.name + " earned $" + str(w.calculatePay(55)) + " for 55
hours.")
```

Write the class along with the constructor and the calculatePay() method.

```
class Worker:

        def __init__(self, name, regular, extended, rate):
            self.name = name;
            self.regular = regular;
            self.extended = extended;
            self.rate = rate;

         def calculatePay(self, hours):
            if hours > self.extended:
                high = hours - self.extended
                over = self.extended - self.regular
                base = self.regular
            elif hours > self.regular:
                over = hours - self.regular
                base = regular
            else:
                base = hours
                over = 0
                high = 0
            pay = (self.regular * self.rate) + (over * self.rate * 1.5) + (high * self.rate * 2);
            return pay
```

11. (Regular Expressions)

a) Following is a regular expression

```
r'\d\d-\d\d-\d\d \d\d:\d\d:\d\d\.\d\d'
```

What will the pattern match from the following text (clearly underline the exact text matched, if any, in each line):

```
18-08-26-08:00:00.01  Start of classes

18-09-23 08:00:00.00  Chuseok starts

18-09-26 23:59:59.99  Chuseok ends

18-12-12-18:30:00.99 - End of classes
```

b) What does the following code print?

```
import re
phone = '123-456-7890'
pattern = r'^\(\d{3}\)-\d{3}-\d{4}$'
if re.search(pattern, phone):
    print('The string matches the pattern.')
else:
    print('The string does not match.')
```

____The string does not match._____

c) After the following code runs, what will be in the variable result?

```
Import re
line = 'the cat and dog'
result = re.sub(r'(.*)(dog)(.*)',
            r'\1mouse\3', line)
```

_______the cat and mouse _________________

12. (Expressions with mod)

What does Python print as the value of the following expressions?

| | |
|---|---|
| 19 % 5 | 4 |
| 21 % 7 | 0 |
| ((21 * 7) + 16) % 31 | 8 |
| ((20 * 80) + 337) % 1000 | 937 |
| ((100 % 19) + 20) % 7 | 4 |
| ((10 * 20) % 5 + 30) % 4 | 2 |
| 17 % 2 + 31 % 2 | 2 |
| (700 + 3) % 70 | 3 |

13. (Code analysis)

a) What does the following code print:

```
contractions = {"I'm": "I am", "You're": "You are", "He's": "He is",
"She's": "She is"}
sentences = ["I'm finished.", "You're good.", "He's there.", "She's
awesome."]
for sentence in sentences:
    words = sentence.split()
    if words[0] in contractions.keys():
        newsentence = contractions[words[0]]
        for word in words[1:]:
            newsentence = newsentence + " " + word
            print(newsentence)
```

I am finished.
You are good.
He is there.
She is awesome.

b) What does the following code print:

```
contractions = {"I'm": "I am", "You're": "You are", "He's": "He is",
"She's": "She is"}
newcont = {contractions[key]:key for key in contractions.keys()}
print(str(newcont))
```

{'I am': "I'm", 'You are': "You're", 'He is': "He's", 'She is': "She's"}