1- What is Computation?

Computation is simply a sequence of steps that can be performed by a computer.

2- Theory of computation

- is the theoretical study of capabilities and limitations of Computers (Develop formal mathematical models of computation that reflect real world computers).
- includes the fundamental mathematical properties of computer hardware, software and their applications. It is a computer science branch which deals with how a problem can be solved efficiently by using an algorithm on a model of computation.

The theory of computation field is divided into three concepts, which are as follows:-

- Automated theory and language.
- Computability theory.
- Complexity theory.

Let us understand these concepts in detail.

1- Automata theory and formal language.

- Which answers - What are computers (Or what are models of computers)
- It deals with the definition and properties of various mathematical models of computers.
- The term automata mean: the plural of automaton and it means" something that works automatically". For example,

- **Finite Automata** − These are used in compilers, hardware design and text processing.
- **Context free grammar** − These are used to define the programming languages and in artificial intelligence.
- **Turing machine** − These are simple abstract models of a real computer.

In automata we will simulates parts of computers. Or we will make mathematical models of computers. Automata are more powerful than any real computer because we can design any machine on papers that can do everything we want.

2- Compatibility theory.

- Which answers - What can be computed by computers?

- Computability theory deals with what can and cannot be computed by the model respectively. The theoretical models are proposed in order to understand the solvable and unsolvable problems which lead to the development of the real computers.

3- Complexity theory.

- Which answers - What can be efficiently computed?
- Complexity theory groups the computable problems based on their hardness. For example,

- Any problem is easy, if it is solved efficiently. For example, sorting sequence, searching name.
- Any problem is hard, if it cannot be solved efficiently. For example, factoring a 500-digit integer into its prime factor.

The main purpose of theory of computation is to develop a formal mathematical model of computation that reflects the real world computers.

3. Applications of Theory of computation

The theory of computation is applied in the following −

- Traffic lights..
- Marketing.
- Compilers.
- Cloud computing.

4. about “Theory of Computation”

The Theory of Computation rests on the fact that computers can’t solve all problems. A given machine would have limitations, and the Theory of Computation aims to discover these. The computational model is given inputs and decides whether or not it can process the information using the developed algorithm.

For example, you can create a machine and design it so that it only accepts red objects. The algorithm is pretty straightforward, as represented by the image below. The red square is accepted, and the model rejects the yellow square.

The Theory of Computation can also help determine if a model needs improvement. In the example above, the developer may want to introduce other inputs to see how the model treats them. What happens when a red square with a yellow border is introduced to the machine? How about when the border is red, but the inside of the object is yellow?

Set Notations

Set: A set is a collection of objects without repetition. Each object in a set is called an element of the set, for example D denotes the set of days of the week:

D = {Sun, Mon, Tue, Wed, Thu, Fri, Sat}

D = {x :x is a day of the week}

If an element x is an element of a set A, then we write $x \in A$ (x belong to A), and if x is not an element of A, we write $x \notin A$ (x does not belongs to A).

Examples in Set Representations:

- C = { a, b, c, d, e, f, g, h, i, j, k }
- C = { a, b, …, k } finite set
- S = { 2, 4, 6, … } infinite set
- S = { j : j > 0, and j = 2k for k>0 }
- S = { j : j is nonnegative and even }
- let A = { 1, 2, 3, 4, 5 } and U means Universal Set: all possible elements
  U = { 1 , … , 10 }

A particular important set is the Empty Set (or null set), this set contains no elements and it's denoted by Φ or { }. Another particular important se is the Universal Set, the elements of all sets under investigation usually belong to the universal set.

المجموعة الشاملة Universal Set : وهي المجموعة التي تحتوي على جميع العناصر التي تخص مجال معين ويرمز لها بالحرف **U** . مثال مجموعة الاحرف هي مجموعة شاملة تضم جميع ابجديات العالم.

المجموعة الخالية Empty Set : وهي المجموعة منعدمة العناصر أي انها لا تحتوي على أي عنصر ويرمز لها بالرمز {} or Φ .

***المجموعة الجزئية* Subset:** هي المجموعة التي تشكل جزءاً من مجموعة أخرى ويرمز لها بالرمز $\subset$

We say a set A is a subset of a set B written $A \subset B$, and if not written $A \not\subset B$:

$\{1,2,4\} \subset \{1,2,3,4,5\}$, and $\{1,2,6\} \not\subset \{1,2,3,4,5\}$

Two sets A and B are said to be equal, written A=B, if A and B contains the same elements.

Basic Operations on Sets

1. Unary operation, e.g. complement.
2. Binary operation, e.g. Union, Intersection, and Difference.

- The complement of a set A, written (A^-, A^c) is the set that contains of all elements in the universal set which are not in A.

$A^c = \{x : x \notin A\}$, contains of all elements in the universal set which are not in A.

- The union of sets A and B, written A∪B
  is a set that contains everything that is in A, or in B, or in both.

$A \cup B = \{x : x \in A \text{ or } x \in B\}$,all elements which belongs to A or B.

- The intersection of sets A and B, written A∩B
  is a set that contains exactly those elements that are in both A and B.

$A \cap B = \{x : x \in A \text{ and } x \in B\}$, all elements which belongs both A and B.

- The set difference of set A and set B, written A – B is a set that contains everything that is in A but not in B.

$A \setminus B = \{x : x \in A \text{ and } x \notin B\}$, all elements which belongs to A but does not belongs to B.

- The length of a set A, written |A|, is the number of elements in a set A .
  ex: A={a,b,c,d} then length of A is 4

- A power set is a set of sets Power : E.g. let S = { a, b, c }
  2^S= { Ø, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c} }
  Observation:| $2^S| = 2^{|S|}$($8 = 2^3$)

P(A) is the power set of A, the set of all subsets of A. (2^A)

Example:

If U= {0,1,2,3,4,…,9}, A= {0,1,3,5}, B= {2,3,5}

Find: A^c, $A \cup B$, $A \cap B$, $A \setminus B$, P(A).

Sol.:

$A \cup B = \{0,1,2,3,5\}$

$A \cap B = \{3,5\}$

$A \setminus B = \{0,1\}$

P(A)= { Φ,{0}, {1}, {3}, {5}, {0,1}, {0,3}, {0,5}, {1,3}, {1,5}, {3,5}, {0,1,3}, {0,1,5}, {0,3,5}, {1,3,5}, {0,1,3,5}}

Some Properties of Sets

Let A, B, and C be subsets of the Universal set U

Associative Laws	$A \cap (B \cap C) = (A \cap B) \cap C$ $A \cup (B \cup C) = (A \cup B) \cup C$
Distributive Properties	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
Idempotent Properties	$A \cap A = A$, $A \cup A = A$
Double Complement Property	$(A^c)^c = A$
De Morgan's Laws	$(A \cup B)^c = A^c \cap B^c$ $(A \cap B)^c = A^c \cup B^c$
Commutative Properties	$A \cap B = B \cap A$ $A \cup B = B \cup A$
Identity Properties	$A \cup \Phi = A$, $A \cap U = A$
Complement Properties	$A \cap A^c = \Phi$, $A \cup A^c = U$

Language

Language is the set of all strings of terminal symbols derivable from alphabet. Alphabet is a finite set of symbols. For example {0, 1} is an alphabet with two symbols, {a, b} is another alphabet with two symbols and English alphabet is also an alphabet. A String (also called a Word) is a finite sequence of symbols of an alphabet.(b, a and aabab) are examples of string over alphabet {a, b}, and 0, 10 and 001 are examples of string over alphabet {0, 1}.A null string is a string with no symbols, usually denoted by epsilon ϵ or lambda (λ) or null Λ.

A languageis a set of strings over an alphabet. Thus {a, ab, baa} is a language (over alphabet {a,b}), and {0, 111} is a language (over alphabet {0,1}). The number of symbols in a string is called the length of the string. For a string w its length is represented by |w|. The empt ystring(also called null string) it has no symbols. The empty string is denoted by (λ),thus $|\lambda| = 0$.

For example: |00100| = 5, |aab| = 3, $|\lambda| = 0$

Rule

Language = alphabet + string (word) + grammar (rules, syntax) + operations on languages (concatenation, union, intersection, Kleene star)

Types of Languages

1- **Talking Language** (e.g.: English, Arabic): It has alphabet: $\sum = \{a, b, c, \ldots, z\}$. From these alphabetic we make sentences that belong to the language. Now we need a grammar to know if this sentence is true or false.

2- **Programming Language**: (e.g.: c++, Pascal):It has alphabetic: $\sum = \{a,b,c,..., z , A,B,C,..., Z, ?, /, - ,\backslash\}$. From these alphabetic we make sentences that belong to programming language. Now we want to know if this sentence is true or false so we need a compiler to make sure that syntax is true.

3- **Formal Language**: (any language we want): It has strings from these strings we make sentences that belong to this formal language. And we want to know if this sentence is true or false so we need rules.

Example:

Alphabetic: ∑ = {0, 1}.

Sentences: 0000001, 1010101.

Rules: Accept any sentence start with zero and refuse sentences that start with one.

So we accept: 0000001 as a sentence satisfies the rules.

And refuse: 1010101 as a sentence doesn't satisfy the rules.

Example:

Alphabetic: ∑ = {a, b}.

Sentences: ababaabb, bababbabb

Rules: Accept any sentence start with (a) and refuse sentences that start with (b).

So we accept: aaaaabba as a sentence satisfies the rules.

And refuse: baabbaab as a sentence doesn't satisfy the rules.

Kleene star, The * Operation (closure)

$\Sigma*$: is the set of all strings obtained by concatenating zero or more strings from Σ

Let $\Sigma = \{a, b\}$

$\Sigma* = \{\lambda, a, b, aa, ab, ba, , bb, aab, \dots\}$

The $^+$ Operation:

$\Sigma+$: the set of all possible strings from alphabet Σ except λ

$\Sigma = \{a, b\}$

$\Sigma* = \{\lambda, a, b, aa, ab, ba, , bb, aaa, aab, \dots\}$

$\Sigma+ = \Sigma* - \lambda$

$\Sigma+ = \{a, b, aa, ab, ba, bb, aaa, aab, \dots\}$

Operations on Languages

1- The usual set operations

$\{a,ab,aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$

$\{a,ab,aaaa\} \cap \{bb, ab\} = \{ab\}$

$\{a,ab,aaaa\} - \{bb,ab\} = \{a,aaaa\}$

2- Complement: $L^- = \Sigma^* - L$

$\Sigma^* = \{\lambda, a, b, aa, ba, ab, bb, aaa, \dots\}$

$\{a,ba\}^- = \{\lambda, b, aa, ab, bb, aaa, \dots\}$

3- Reverse

Definition: $L^R = \{w^R : w \in L\}$

Examples: $\{ab,aab,baba\}^R = \{ba,baa,abab\}$

$L = \{a^n b^n : n \geq 0\}$

$L^R = \{b^n a^n : n \geq 0\}$

4- Concatenation

Definition: $L1L2 = \{xy : x \in L1, y \in L2\}$

Example: $\{a,ab,ba\}\{b,aa\} = \{ab,aaa,abb,abaa,bab,baaa\}$

5- Another operation

Definition: $L^n = \underbrace{LL\ldots L}_{n}$

$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} = \{aaa,aab,aba,abb,bab,baa,bba,bbb\}$

Special case: $L^0 = \{\lambda\}$

$\{a,bba,aaa\}^0 = \{\lambda\}$

6- Star-closure (Kleen*)

Definition: $L^* = L^0 \cup L^1 \cup L^2 \ldots$

Example:

$$\{a,bb\}^* = \begin{Bmatrix} \lambda, \\ a,bb, \\ aa,abb,bba,bbbb, \\ aaa,aabb,abba,abbbb,\ldots \end{Bmatrix}$$

7- positive-closure

Definition: $L^+ = L^1 \cup L^2 \ldots$

$= L^* - \{\lambda\}$

$$\{a,bb\}^+ = \begin{Bmatrix} a,bb, \\ aa,abb,bba,bbbb, \\ aaa,aabb,abba,abbbb,\ldots \end{Bmatrix}$$

7- positive-closure

Definition: $L^+ = L^1 \cup L^2 \ldots$

$= L^* - \{\lambda\}$

$$\{a,bb\}^+ = \begin{Bmatrix} a,bb, \\ aa,abb,bba,bbbb, \\ aaa,aabb,abba,abbbb,\ldots \end{Bmatrix}$$

Example: Consider the languages L1 = { λ , 0, 1} and L2 = { λ, 01, 11}.

- The union of these languages is L1∪ L2 = { λ , 0, 1, 01, 11}
- Their intersection is L1 ∩ L2 = { λ }
- And the complementation of :

$L1^- = U - L1$

$= \{\lambda,0,1,00,01,10,11,000,001,\ldots\} - \{\lambda,0,1\}$

$= \{00, 01, 10, 11, 000, 001, \ldots\}.$

8- Palindrome: is a language that={Λ, all string x such that reverse(x)=x}

Example: aba,aabaa,bab,bbb,…

Regular Expression

Is a set of symbols, thus if alphabet= {a, b}, then aab, a, baba, bbbbb, and baaaaa would all be strings of symbols of alphabet. In addition we include an empty string denoted by λ which has no symbols in it.

Examples of Kleene star (Kleene closure):

1* is the set of strings {λ, 1, 11, 111, 1111, 11111, …, etc.}

(1100)* is the set of strings {λ, 1100, 11001100, 110011001100, …, etc.}

(00+11)* is the set of strings {λ, 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, …, etc.}

(0+1)* is all possible strings of zeros and ones, often written as ∑* where ∑= {0, 1}.

(0+1)*(00+11) is all strings of zeros and ones that end with either 00 or 11.

$(w)^+$ is a shorthand for (w)(w)*, w is any string or expression with the superscript (+).

Concatenation:

Notation to the concatenation is (.) (The dot):

IfL1 = {x, xxx} and L2 = {xx}, so (L1.L2) means L1 concatenated L2 and it is equal to {xxx, xxxxx}

Examples on concatenations:

Ex1:

L1 = {a, b}.
L2 = {c, d}.
L1.L2 = {ac, ad, bc, bd}

Ex2:

∑= {x}.
L1 = {set of all odd words over _ with odd length}.

L1 = {set of all even words over _ with odd length}.
L1= {x, xxx, xxxxx, xxxxxxx……}.
L2= {_, xx, xxxx, xxxxxx…}.
L1.L2 = {x, xxx, xxxxx, xxxxxxx…}.

التكرار غير مسموح داخل المجموعة

Ex3:

L1 = {x, xxx}.
L2 = {xx}.
L1.L2 = {xxx, xxxxx}.

Some rules on concatenation

λ.x = x

L1.L2 = {set of elements}

Definition of a Regular Expression

A regular expression may be the null string, r = λ
A regular expression may be an element of the input alphabet, r = a
A regular expression may be the union of two regular expressions, $r = r_1 + r_2$
A regular expression may be the concatenation of two regular expressions, $r = r_1 r_2$
A regular expression may be the Kleene closure (star) of a regular expression $r = r_1^*$
A regular expression may be a regular expression in parenthesis $r = (r_1)$

λ is the zero length string
0, 1, a, b, c, are symbols in ∑
x is a variable or regular expression
(...)(...) is concatenation
(...) + (...) is union
(...)* is the Kleene Closure = Kleene Star

(λ)(x) = (x)(λ) = x

$(\lambda) + (x) = (x) + (\lambda) = x$
$x + x = x$
$(\lambda)^* = (\lambda)(\lambda) = \lambda$
$(x)^* + (\lambda) = (x)^* = x^*$
$(x + \lambda)^* = x^*$
$x^* (a+b) + (a+b) = x^* (a+b)$
$x^* y + y = x^* y$
$(x + \lambda)x^* = x^* (x + \lambda) = x^*$
$(x+ \lambda)(x+ \lambda)^* (x+ \lambda) = x^*$

λ is the null string (there are no symbols in this string)

* is the set of all strings of length greater than or equal 0.

Example:
A = {a,b} // the alphabet is composed of a and b
$A^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$
The symbol * is called the Kleene star.
Φ (empty set)
λ (empty string)
( ) delimiter,
$\cup$ + union (selection)
$\cap$ × intersection
Concatenation

Given regular expressions x and y, x + y is a regular expression
Representing the set of all strings in either x or y (set union)
x = {a, b}, y = {c, d}, x + y = {a, b, c, d}

Example
Let A={0,1}, W_1 = 00110011, W_2 = 00000
W_1W_2 = 0011001100000
W_2W_1 = 0000000110011
$W_1\lambda = W_1$ = 00110011

$\lambda W_2 = W_2 = 00000$

Example

x = {a, b}, y = {c, d}, x.y = {ac, ad, bc, bd}

Note: (a + b)* = (a* b*)*

Examples of Regular Expressions (RE)

Describe the language = what is the output (words, strings) of the following RE

Regular Expression	Output (Set of Strings)
λ	{ λ}
λ*	{ λ}
a	{a}
aa	{aa}
a*	{ λ, a, aa, aaa, …}
aa*	{a, aa, aaa, …}
a+	{a, aa, aaa, …}
ab*	{a, ab, abb, abbb,…}
(ab)*	{ λ, ab, abab, ababab,… }
ba+	{ba, baa, baaa, …}
(ba)+	{ba, baba, bababa, …}
(a | b)	{a , b}
a | b *	{a, λ, b, bb, bbb, …}
(a | b) *	{λ, a, b, aa, ab, ba, bb, …}
a(ba)*b	{ ab, abab, ababab, abababab,…}
aa(ba)*bb	{aabb, aababb, aabababb,…}
(a+a)	{a}
(a+b)	{a, b}
$(a + b)^2$	(a + b)(a + b)={aa, ab, ba, bb}
(a + b + c)	{a, b, c}
(a + b)*	{λ, a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, …}
(abc)	{abc}
(λ + a) bc	{bc, abc}

a + b*	{a, λ, b, bb, bbb, …}
a *+ b*	{ λ ,a, b, aa, bb, aaa, bbb,…}
a (a + b)*	{a, aa, ab, aaa, abb, aba, abaa, …}
(a + b)* a (a + b)*	{a, aaa, aab, baa, bab, …}
(a+b)+(a+b)	{a,b}
(a + λ)*	(a)* = {λ, a, aa, aaa, …}
a(λ)*b	{ab}
x*(a+b)	{a, b, xa, xb, xxa, xxb, xxxa, xxxb,…}
x* (a + b) + (a + b)	x* (a + b)
x* y + y	x* y
(x + λ) x*	x* (x + λ) = x*
(x + λ) (x + λ)* (x + λ)	x*
start with a	a (a + b)*
end with b	(a + b)* b
start with a and end with b	a (a+b)* b
start with a or b	(a+b) (a+b)*
end with 00 or 01	(0+1)* (00+01)
end with ab	(a+b)* ab
contains exactly 2 a's	(b)* a (b)* a (b)*
contains at least 2 a's	(a + b)* a (a + b)* a (a + b)*
contains exactly 2 a's or 2 b's	[(b)* a (b)* a(b)*] + [(a)* b (a)* b (a)*]
contains even no. of a	[(b)* a (b)* a (b)*]*
with even length of a	(aa)+

H.W.

Q.1: Find a regular expression over the alphabet {a, b} that contains exactly three a's.

Q.2: Find a regular expression over the alphabet {a, b} that end with ab.

Q.3: Find a regular expression over the alphabet {a, b} that has length of 3.

Q.4: Find a regular expression over the alphabet {a, b} that contain exactly two successive a's.

Q.5: Find the output strings (output words) for the following regular expressions:

Regular expression	Output Strings
(λ)*	
(x)* + (λ)	

aa* b	
baa* a	
(a + b)* ba	
(0 + 1)* 00 (0 + 1)*	
(11 + 0)* (0 + 11)*	
01* + (00 + 101)*	
(a + b)* abb+	
(((01 + 10)* 11)* 00)*	