Padasalai's Telegram Groups!

(தலைப்பிற்கு கீழே உள்ள லிங்கை கிளிக் செய்து குழுவில் இணையவும்!)

- Padasalai's NEWS - Group
https://t.me/joinchat/NIfCqVRBNj9hhV4wu6_NqA

- Padasalai's Channel - Group
https://t.me/padasalaichannel

- Lesson Plan - Group
https://t.me/joinchat/NIfCqVWwo5iL-21gpzrXLw

- 12th Standard - Group
https://t.me/Padasalai_12th

- 11th Standard - Group
https://t.me/Padasalai_11th

- 10th Standard - Group
https://t.me/Padasalai_10th

- 9th Standard - Group
https://t.me/Padasalai_9th

- 6th to 8th Standard - Group
https://t.me/Padasalai_6to8

- 1st to 5th Standard - Group
https://t.me/Padasalai_1to5

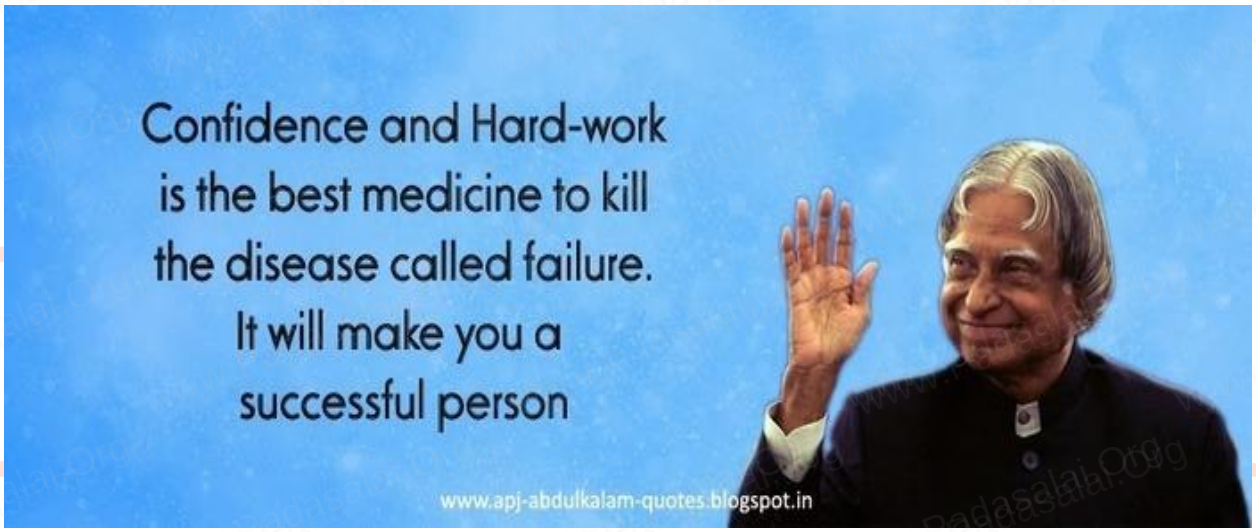
- TET - Group
https://t.me/Padasalai_TET

- PGTRB - Group
https://t.me/Padasalai_PGTRB

- TNPSC - Group
https://t.me/Padasalai_TNPSC

XII – COMPUTER SCIENCE

JUNE MONTHLY TEST STUDY MATERIALS



NAME :

CLASS :

PREPARED BY

M.DHANAPAL,MCA,B.Ed.,

LITERACY MISSION MATRICULATION HIGHER SECONDARY SCHOOL

CHAPTER – 1 FUNCTION

CHOOSE THE CORRECT ANSWER:

1.Which of the following is important criteria complete the task?
a) Program b) code c) **algorithm** d) pseudo code

2. The duration of computation time must be independent of
a) Compiler b) pseudo code
c) Programming language d) **a & c**

3. The algorithms are expressed using_____of a programming language.
a) Functions b) subroutines c) **statements** d) reference

4.If a bulk of statements to be repeated for many no.of times then_____are used to finish the task.
a) **subroutines** b) programs c) required d) statements

5._ is the basic building blocks of computer programs.
a) Programs b) function c) pure function d) **Subroutines**

6. _____ are small sections of code that are used to perform a particular task that
can be used repeatedly.
a) Impure function b) **subroutines**
c) Pure function d) programming language

7.In Programming languages these subroutines are called as_____.
a) Subroutines b) **Functions** c) pseudo code d) Inference

8. A _____ is a unit of code that is often defined within a greater code structure.
a) Routine b) Node c) **Function** d) Program

9._ is distinct syntactic blocks.
a) **Definitions** b) Declaration c) Statement d) Required

10. _____ is the variables in a function definition.
a) Algorithms b) programs c) **Parameters** d) Functions

11. is the values which are passed to a function definition.
a) **Arguments** b) impurefunction c) statement d) algorithm

12. There are types of parameters are in the functions.
a) 4 b) **2** c) 3 d) 5

13. The syntax for function definition is_____.
a) **let rec fn a1 a2 ... an := k** b) let receive fn a1 a2 ... an := k
c) let rec function a1 a2 ... an := k d) let rec fn a1 a2 ... an :> k

14. The keyword____is required if 'fn' is to be a recursive function; otherwise it may be omitted.
a) record b) receive c) **rec** d) recover

15. A function definition which call itself is called_____function.
a) record b) **recursive**
c) return d) destroy

16. All functions are_____definitions.
a) single b) dynamic c) **static** d) dual

17. An_____is a set of action that an object can do.
a) interconnect b) interflow c) function d) **interface**

18. _ are functions which will give exact result when the same arguments are passed.
a) **Pure functions** b) inner function
c) outer function d) impure function

19. The another name of side - effect_____.
a) Pure functions b) inner function
c) outer function d) **impure function**

20. The return value of the_____solely depends on its arguments passed.
a) **Pure functions** b) inner function
c) outer function d) impure function

21. _____ contains a set of code that works on many kinds of inputs and produces a concrete output.
a) interconnect b) interflow c) **function** d) interface

22. When you write the type annotation the_____are mandatory in the function definition.
a) set braces b) **parentheses** c) slashes d) dots

23. carries out the instructions defined in the interface.
a) Abstraction b) Reduction c) Collusion d) **Implementation**

24. There are characteristics have in interface.
a) 3 b) 4 c) 5 d) **2**

25. ***let rec fna1 a2 ... an := k*** in this ***'fn'*** indicating the_____of the function name.
a) String b) Character c) **Identifier** d) Constant

26. _____ do not modify the arguments which are passed to them.
a) inner function b) **Pure functions**
c) outer function d) impure function

27. _____ may modify the arguments which are passed to them.
a) inner function b) Pure functions
c) outer function d) **Impure function**

28. One of the most popular groups of side effects is modifying the variable _____ of function.
a) **outside** b) topside c) inside d) None of these

29. ***let y: = 0***
(int) inc (int) x
y: = y + x;
return (y)
In the above Algorithm function. The side effects of the_____function is it is changing the data of the external visible variable____.
a) gcd( ), x b) int( ), y c) **inc (), y** d) inc (), x

30. An object's_____and_____is controlled by sending functions to the object.
a) **attributes, behaviour** b) function, behaviour
c) function, attributes d) None of these

2-Marks

1. What is subroutine?

Subroutines are the basic building blocks of computer programs. Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly.

2. Define Algorithm.

Algorithms are expressed using statements of a programming language.

3. Define Function with respect to Programming language.

A function is a unit of code that is often defined within a greater code structure. Specifically, a function contains a set of code that works on many kinds of inputs, like variants, expressions and produces a concrete output.

4. Write the inference you get from X:=(78).

In the above function defnition if expression can return ***1*** in the then branch, by the ***typing*** rule the entire if expression has type ***int***. We get inference of expression is **int**.

5. Differentiate interface and implementation.

Interface	Implementation
Interface just defines what an object can do, but won't actually do it.	Implementation carries out the instructions defined in the interface.

6. Which of the following is a normal function definition and which is recursive function defnition.

i) let rec sum x y:
return x + y
ii) let disp :
print 'welcome'
iii) let rec sum num:
if (num!=0) then return num + sum (num-1)
else
return num

(i) Recursive function definition (ii) Normal function definition
(iii) Recursive function definition

3-Marks

7. Mention the Characteristics of interface.

Characteristics of interface

• The class template specifies the interfaces to enable an object to be created and operated properly.

• An object's attributes and behavior is controlled by sending functions to the object.

8. Why strlen is called pure function?

strlen is a pure function because the function takes one variable as a parameter, and accesses it to find its length. This function reads external memory but does not change it, and the value returned derives from the external memory accessed.

9. What is the side effect function of impure function? Give example.

The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function.

For example

The mathematical function random( ) will give different outputs for the same function call.

```
let Random number
let a := random()
if a > 10 then
return: a
else
return: 10
```

10. Differentiate between Pure function and Impure function.

Pure Function	Impure Function
The return value of the pure functions solely depends on its arguments passed. Hence, if you call the pure functions with the same set of arguments, you will always get the same return values. They do not have any side effects.	The return value of the impure functions does not solely depend on its arguments passed. Hence, if you call the impure functions with the same set of arguments, you might get the different return values. For example, random(), Date().
They do not modify the arguments which are passed to them.	They may modify the arguments which are passed to them.

11. What happens if you modify a variable outside the function? Give an example.

Modify variable outside a function

One of the most popular groups of side effects is modifying the variable outside of function. For example

```
let y: = 0
(int) inc (int) x
y: = y + x;
return (y)
```

In the above example the value of y get changed inside the function defnition due to which the result will change each time. The side effects of the inc () function is it is changing the data of the external visible variable ***'y'***. As you can see some side effects are quite easy to spot and some of them may tricky. A good sign that our function impure *(has side effects)* is that it doesn't take any arguments and it doesn't return any value.

5 - Marks

12. What are called Parameters and write a note on
(i) Parameter without Type (ii) Parameter with Type

Parameters (and arguments)

Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.

1. Parameter without Type

Let us see an example of a function definition:

```
(requires: b>=0 )
(returns: a to the power of b) let rec pow a b:=
        if b=0 then 1
        else a * pow a (b-1)
```

In the above function definition variable *'b'* is the parameter and the value which is passed to the variable *'b'* is the argument. Te precondition ***(requires)*** and post condition ***(returns)*** of the function is given. Note we have not mentioned any types: (***data types***). Some language compiler solves this type (***data type***) inference problem algorithmically, but some require the type to be mentioned.

In the above function definition if expression can return ***1*** in the then branch, by the ***typing*** rule the entire if expression has type ***int***. Since the if expression has type ***'int',*** the function's return type also be *'int'*. ***'b'*** is compared to *0* with the equality operator, so ***'b'*** is also a type of *'int'*. Since ***'a'*** is multiplied with another expression using the * operator, ***'a'*** must be an int.

2. Parameter with Type

Now let us write the same function definition with types for some reason:

```
(requires: b> 0 )
(returns: a to the power of b )
        let rec pow (a: int) (b: int) : int :=
                if b=0 then 1
        else a * pow b  (a-1)
```

When we write the type annotations for ***'a'*** and ***'b'*** the parentheses are mandatory. Generally we can leave out these annotations, because it's simpler to let the compiler infer them. There are times we may want to explicitly write down types. This is useful on times when you get a type error from the compiler that doesn't make sense. Explicitly annotating the types can help with debugging such an error message. The syntax to define functions is close to the mathematical usage: the definition is introduced by the keyword ***let***, followed by the ***name*** of the function and its *arguments*; then the formula that computes the image of the argument is written after an = sign. If you want to define a recursive function: use ***"let rec" instead of "let".***

Syntax:

The syntax for function definitions:

```
let rec fna1 a2 ... an := k
```

Here the ***'fn'*** is a variable indicating an identifier being used as a function name. The names ***'a1'*** to ***'an'*** are variables indicating the identifiers used as parameters. The keyword ***'rec'*** is required if ***'fn'*** is to be a recursive function; otherwise it may be omitted.

13. Identify in the following program

let rec gcd a b :=
if b <> 0 then gcd b (a mod b) else return a

i) Name of the function
ii) Identify the statement which tells it is a recursive function
iii) Name of the argument variable
iv) Statement which invoke the function recursively
v) Statement which terminates the recursion

(i) gcd( ) function
(ii) recursively called till the variable *'b'* becomes *'0'*
(iii) *b* and *(a mod b)* are two arguments passed to *'a'* and *'b'* of the gcd function.
(iv) *(a mod b) until 'b' became '0'.*
(v) return a. or (When variable 'b' became '0' terminated).

14. Explain with example Pure and impure functions.

PURE FUNCTIONS

Pure functions are functions which will give exact result when the same arguments are passed. For example the mathematical function sin (0) always results **0**. This means that every time you call the function with the same arguments, you will always get the same result. A function can be a pure function provided it should not have any external variable which will alter the behaviour of that variable. Let us see an example

let square x
return: x * x

The above function square is a pure function because it will not give different results for same input. There are various theoretical advantages of having pure functions. One advantage is that if a function is pure, then if it is called several times with the same arguments, the compiler only needs to actually call the function once. Let's see an example

let i: = 0;
if i <strlen (s) then
-- Do something which doesn't affects
++i

If it is compiled, ***strlen (s)*** is called each time and strlen needs to iterate over the whole of ***'s'.*** If the compiler is smart enough to work out that strlen is a pure function and that ***'s'*** is not updated in the loop, then it can remove the redundant extra calls to strlen and make the loop to execute only one time. From these what we can understand, strlen is a pure function because the function takes one variable as a parameter, and accesses it to find its length. This function reads external memory but does not change it, and the value returned derives from the external memory accessed.

IMPURE FUNCTIONS

The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function. When a function depends on variables or functions outside of its defnition block, you can never be sure that the function will behave the same every time it's called. For example the mathematical function random() will give different outputs for the same function call.

```
let Random number
let a := random()
if a > 10 then
return: a
else
return: 10
```

Here the function Random is impure as it is not sure what will be the result when we call the function.

15. Explain with an example interface and implementation.

INTERFACE VS IMPLEMENTATION

An interface is a set of action that an object can do. For example when you press a light switch, the light goes on, you may not have cared how it splashed the light. In Object Oriented Programming language, an Interface is a description of all functions that a class must have in order to be a new interface. In our example, anything that ***"ACTS LIKE"*** a light, should have function definitions like turn_on () and a turn_off (). The purpose of interfaces is to allow the computer to enforce the properties of the class of ***TYPE T*** *(whatever the interface is)* must have functions called *X, Y, Z,* etc.

A class declaration combines the external interface *(its local state)* with an implementation of that interface *(the code that carries out the behaviour)*. An object is an instance created from the class. The interface defines an object's visibility to the outside world.

In object oriented programs classes are the interface and how the object is processed and executed is the implementation.

The person who drives the car doesn't care about the internal working. To increase the speed of the car he just presses the accelerator to get the desired behaviour. Here the accelerator is the interface between the driver *(the calling / invoking object)* and the engine *(the called object)*.

In this case, the function call would be Speed (70): This is the interface. Internally, the engine of the car is doing all the things. It's where fuel, air, pressure, and electricity come together to create the power to move the vehicle. All of these actions are separated from the driver, who just wants to go faster. Thus we separate interface from implementation. Let us see a simple example, consider the following implementation of a function that finds the minimum of its three arguments:

```
let min 3 x y z :=
        if x < y then
                if x < z then x else z
        else
                if y < z then y else z
```

CHAPTER – 2 DATA ABSTRACTION

CHOOSE THE CORRECT ANSWER:

1. _____ is a powerful concept in computer science that allows programmers to treat code as objects.
   a) Specification b) **Data abstraction** c) Constructor d) Selector
2. _____ means splitting a program in to many modules.
   a) Object b) Collusion c) **Modularity** d) Selector
3. _____ is a type or class for objects whose behavior is defined by a set of value and a set of operations.
   a) **Abstract Data Type** b) Abstract Defined Type
   b) Added Data Type d) Abstract Data Table
4. Which gives an implementation independent view?
   a) Constructor b) **Abstract** c) Collusion d) None of these
5. The process of providing only the essentials and hiding the details is known as_____.
   a) Definition b) Constructor c) Selector d) **Abstraction**
6. _____ and_____ADT can be implemented using lists.
   a) Stack, Row b) Row, Queue c) **Stack, Queue** d) Cols, Row
7. To facilitate data abstraction, you will need to create types of functions.
   a) 4 b) **2** c) 5 d) more
8. To facilitate data abstraction, you will need to create functions are and_____.
   a) Abstract, Selector b) Constructor, Stack
   c) Constant, Selector d) **Constructor, Selector**
9. _____ are functions that build the abstract data type.
   a) Stack b) Selector c) **Constructor** d) Abstraction
10. _____ are functions that retrieve information from the data type.
    a) **Selector** b) Constructor c) Definition d) ADT
11. The data structure which is a mutable ordered sequence of elements is called _____.
    a) Built-in b) **List** c) Tuple d) Pair
12. A sequence of immutable objects is called
    a) Built in b) List c) **Tuple** d) Derived data
13. The data type whose representation is unknown are called_____data type.
    a) Built-in b) Derived c) Concrete d) **Abstract**
14. The data type whose representation is known are called_____data type.
    a) Built-in b) Derived c) **Concrete** d) Abstract
15. Which of the following is a compound structure?
    a) **Pair** b) Triples c) Tuple d) Squared
16. A rational number typically written as_____.
    a) **<numerator>/<denominator>** b) <numerator>/<divider>
    c) <regulator>/<denominator> d) <numerator>/<decimal>
17. A powerful strategy for designing programs_____.
    a) Wishful Thought b) Wonder Thinking
    c) **Wishful Thinking** d) None
18. List is constructed by placing expressions within _____ separated by _____.
    a) ( ) and ; b) ( ) and , c) [ ] and ; d) **[ ] and ,**

19. Any way of bundling two values together into one can be considered as _____.
    a) **Pair** b) List c) Tuple d) Table
20. List can be called as_____.
    a) Tuples b) Table c) **Pairs** d) Command
21. A tuple is a_____separated sequence of values surrounded with_____.
    a) **, and ( )** b) ; and [ ] c) : and( ) d) ? and ( )
22. The elements of a list can be accessed in_____ways.
    a) Four b) One c) Five d) **Two**
23. _____ is a compound structure which is made up of list or Tuple.
    a) **Pair** b) Triples c) Definition d) Squared
24. _____ does not allow to name the various parts of a multi-item object.
    a) Tuple b) **List** c) Pair d) All of these
25. We can define_____as bundled data and the_____that work on that data.
    a) Class, Tuple b) Tuple, List c) Class, Pair d) **Class, Functions**

2-Marks

1. What is abstract data type?

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.

2. Define Data Abstraction. Give Example.

Data abstraction is a powerful concept in computer science that allows programmers to treat code as objects.
For example: car objects, pencil objects, people objects, etc.

3. Definition of ADT.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.

It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.

It is called "abstract" because it gives an implementation independent view.

4. What is meant by abstraction?

The process of providing only the essentials and hiding the details is known as abstraction.

5. What is the ways of implement the ADT?

There can be different ways to implement an ADT, for example, the List ADT can be implemented using singly linked list or doubly linked list. Similarly, stack ADT and Queue ADT can be implemented using lists.

6. What are all functions to be created for facilitate data abstraction?

To facilitate data abstraction, you will need to create two types of functions. They are (i) constructors (ii) selectors.

7. Differentiate constructors and selectors.

Constructors	Selectors
Constructors are functions that build the abstract data type.	Selectors are functions that retrieve information from the data type.
city = makecity(name, lat, lon) Here, makecity(name, lat, lon) is the constructor which creates the object city.	**getname(city)** **getlat(city)** **getlon(city)** are the selectors because these functions extract the information of the city object.

8. hat is rational number? Give Example.

A rational number is a ratio of integers, and rational numbers constitute an important sub-class of real numbers. A rational number such as 8/3 or 19/23 is typically written as:
<numerator>/<denominator>

9. What is Pair?

Pair is a compound structure which is made up of list or Tuple. Bundling two values together into one can be considered as a pair.

10. What is List? Give Example.

List is constructed by placing expressions within square brackets separated by commas. Such an expression is called a list literal. List can store multiple values. Each value can be of any type and can even be another list.

Example for List is [10, 20].

11. What is Tuple? Give Example.

A tuple is a comma-separated sequence of values surrounded with parentheses. Tuple is similar to a list.

Example colour= ('red', 'blue', 'Green')

12. What is difference between List and Tuple?

The difference between the two is that you cannot change the elements of a tuple once it is assigned whereas in a list, elements can be changed.

13. How can we access the elements of list?

The elements of a list can be accessed in two ways.

- The first way is via our familiar method of multiple assignment.
- A second method for accessing the elements in a list is by the element selection operator, also expressed using square brackets.

3-Marks

14. What are the representation of Abstract Data Type Using Rational numbers? (or) Differentiate between Abstract data and Concrete Data.

A concrete data type is a data type whose representation is known and in abstract data type the representation of a data type is unknown.

The part that operates on abstract data and the part that defines a concrete representation.

15. Which strategy is used for program designing? Define that Strategy.

We are using here a powerful strategy for designing programs: 'Wishful Thinking'.

Wishful Thinking is the formation of beliefs and making decisions according to what might be pleasing to imagine instead of by appealing to reality.

16. Identify Which of the following are constructors and selectors?

(a) N1=number() (b) accetnum(n1) (c) displaynum(n1)
(d) eval(a/b) (e) x,y= makeslope (m), makeslope(n)
(f) display()

a) Constructor b) Selector c) Selector
d) Selector e) Constructor f) Selector

17. What are the different ways to access the elements of a list. Give example.

The elements of a list can be accessed in two ways. The first way is via our familiar method of multiple assignment, which unpacks a list into its elements and binds each element to a different name.

```
lst := [10, 20]
x, y := lst
```

In the above example ***x*** will become10 and ***y*** will become 20.

A second method for accessing the elements in a list is by the element selection operator, also expressed using square brackets. Unlike a list literal, a square brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.

```
lst[0]
10
lst[1]
20
```

In both the example mentioned above mathematically we can represent list similar to a set.

18. Identify Which of the following are List, Tuple and class ?

(a) arr [1, 2, 34] (b) arr (1, 2, 34) (c) student [rno, name, mark]
(d) day= ('sun', 'mon', 'tue', 'wed') (e) x= [2, 5, 6.5, [5, 6], 8.2]
(f) employee [eno, ename, esal, eaddress]

a) List b) Tuple c) Class
d) Tuple e) List f) Class

(If Any mistake ignore Answer/ me)

5 - Marks

19. How will you facilitate data abstraction. Explain it with suitable example.

To facilitate data abstraction, you will need to create two types of functions: constructors and selectors.

- Constructors are functions that build the abstract data type.
- Selectors are functions that retrieve information from the data type.

For example, say you have an abstract data type called city. This city object will hold the city's name, and its latitude and longitude. To create a city object, you'd use a function like

city = makecity (name, lat, lon)

To extract the information of a city object, you would use functions like

- **getname(city)**
- **getlat(city)**
- **getlon(city)**

The following pseudo code will compute the distance between two city objects:

```
distance(city1, city2):
lt1, lg1 := getlat(city1), getlon(city1)
lt2, lg2 := getlat(city2), getlon(city2)
return ((lt1 - lt2)**2 + (lg1 - lg2)**2))1/2
```

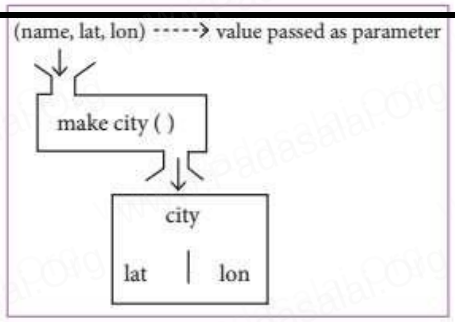
In the above code read distance(), getlat() and getlon() as functions and read lt as latitude and lg longitude. Read := as "assigned as" or "becomes"

lt1, lg1 := getlat(city1), getlon(city1)

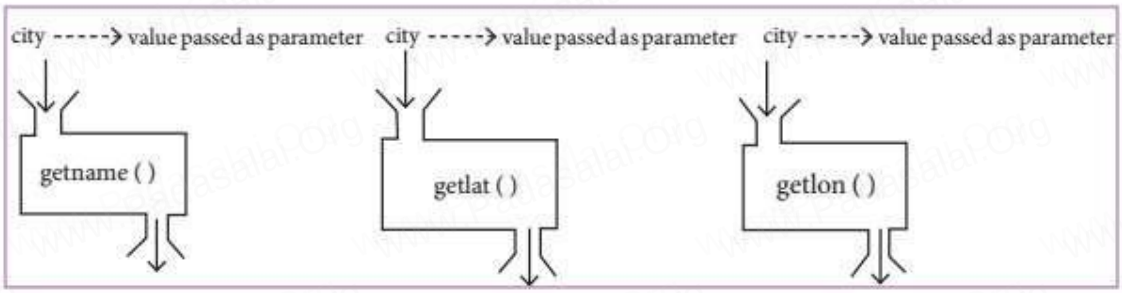
is read as lt1 becomes the value of getlat(city1) and lg1 becomes the value of getlont(city1).

Let us identify the constructors and selectors in the above code As you already know that Constructors are functions that build the abstract data type. In the above pseudo code the function which creates the object of the city is the constructor.

city = makecity (name, lat, lon)



Here makecity (name, lat, lon) is the constructor which creates the object city.



Selectors are nothing but the functions that retrieve information from the data type. Therefore in the above code

- **getname(city)**
- **getlat(city)**
- **getlon(city)**

are the selectors because these functions extract the information of the city object

Now let us consider one more example to identify the constructor and selector for a slope.Read - - as comments.

```
- - constructor
makepoint(x, y):
return x, y
- - selector
xcoord(point):
return point[0]
- -selector
ycoord(point):
return point[1]
```

20. What is a List? Why List can be called as Pairs. Explain with suitable example.

List is constructed by placing expressions within square brackets separated by commas. Such an expression is called a list literal. List can store multiple values. Each value can be of any type and can even be another list.

Example for List is [10, 20].

The elements of a list can be accessed in two ways. The first way is via our familiar method of multiple assignment, which unpacks a list into its elements and binds each element to a different name.

lst := [10, 20]
x, y := lst

In the above example **x** will become10 and **y** will become 20.

A second method for accessing the elements in a list is by the element selection operator, also expressed using square brackets. Unlike a list literal, a square brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.

lst[0]
10
lst[1]
20

In both the example mentioned above mathematically we can represent list similar to a set.

lst[(0, 10), (1, 20)] – where

Any way of bundling two values together into one can be considered as a pair. Lists are a common method to do so. Therefore List can be called as Pairs.

Representing Rational Numbers Using List

You can now represent a rational number as a pair of two integers in pseudo code : a numerator and a denominator.

rational(n, d):
return [n, d]
numer(x):
return x[0]
denom(x):
return x[1]

21. How will you access the multi-item. Explain with example.

As you already know that List allow data abstraction in that you can give a name to a set of memory cells. For instance, in the game Mastermind, you must keep track of a list of four colors that the player guesses. Instead of using four separate variables (color1, color2, color3, and color4) you can use a single variable 'Predict',

e.g., Predict =['red', 'blue', 'green', 'green']

What lists do not allow us to do is name the various parts of a multi-item object. In the case of a Predict, you don't really need to name the parts: using an index to get to each color suffices.

But in the case of something more complex, like a person, we have a multi-item object where each 'item' is a named thing: the firstName, thelastName, the id, and the email. One could use a list to represent a person:

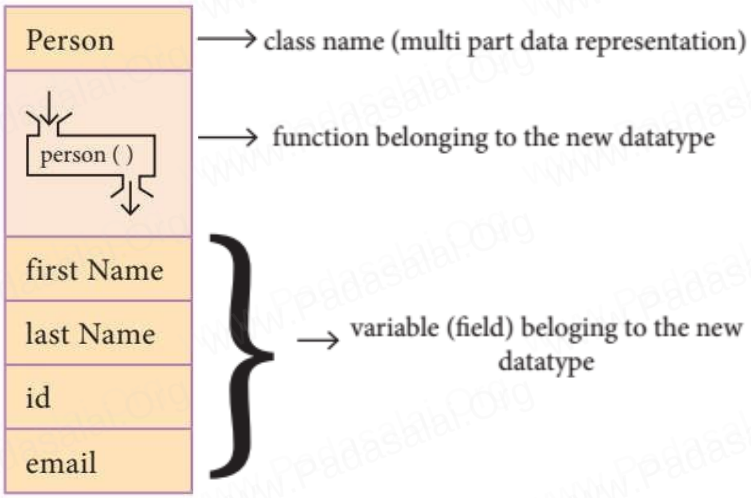
person=['Padmashri', 'Baskar', '994- 222-1234', 'compsci@gmail.com']

but such a representation doesn't explicitly specify what each part represents.

For this problem instead of using a list, you can use the structure construct (In OOP languages it's called class construct) to represent multi-part objects where each part is named (given a name). Consider the following pseudo code: class Person:

```
class Person
    person( )
    firstName := " "
    lastName := " "
    id := " "
    email := "  "
```

The new data type Person is pictorially represented as



Let main() contains	
p1:=Person()	statement creates the object.
firstName := " Padmashri "	setting a field called firstName with value Padmashri
lastName :="Baskar"	setting a field called lastName with value Baskar
id :="994-222-1234"	setting a field called id value 994-222-1234
email="compsci@gmail.com"	setting a field called email with value compsci@gmail.com
- - output of firstName : Padmashri	

The class (structure) construct defines the form for multi-part objects that represent a person. Its defnition adds a new data type, in this case a type named Person. Once defined, we can create new variables (instances) of the type.

In this example Person is referred to as a class or a type, while p1 is referred to as an object or an instance. You can think of class Person as a cookie cutter, and p1 as a particular cookie. Using the cookie cutter you can make many cookies. Same way using class you can create many objects of that type. So far, you've seen how a class defines a data abstraction by grouping related data items.

A class is not just data, it has functions defined within it. We say such functions are subordinate to the class because their job is to do things with the data of the class, e.g., to modify or analyze the data of a Person object.

Therefore we can define a class as bundled data and the functions that work on that data.

CHAPTER – 3 SCOPING

CHOOSE THE BEST ANSWER

1. Which of the following refers to the visibility of variables in one part of a program to another part of the same program.
a) Scope b) Memory c) Address d) Accessibility

2. The process of binding a variable name with an object is called
a) Scope **b) Mapping** c) late binding d) early binding

3. Which of the following is used in programming languages to map the variable and object?
a) :: b) **:=** c) = d) ==

4. Containers for mapping names of variables to objects is called
a) Scope b) Mapping c) Binding **d) Namespaces**

5. Which scope refers to variables defined in current function?
a) Local Scope b) Global scope c) Module scope d) Function Scope

6. The process of subdividing a computer program into separate sub-programs is called
a) Procedural Programming **b) Modular programming**
c) Event Driven Programming d) Object oriented Programming

7. Which of the following security technique that regulates who can use resources in a computing environment?
a) Password b) Authentication **c) Access control** d) Certification

8. Which of the following members of a class can be handled only from within the class?
a) Public members b) Protected members
c) Secured members **d) Private members**

9. Which members are accessible from outside the class?
a) Public members b) Protected members
c) Secured members d) Private members

10. The members that are accessible from within the class and are also available to its sub-classes is called
a) Public members **b) Protected members**
c) Secured members d) Private members

11. Variables are____to an object in memory.
a. addresses b. references c. pointers **d. Either A or B or C**

12. The____of a variable is that part of the code where it is visible.
a. scope b. access specifiers
c. address d. None of these

13. The____rule is used to decide the order in which the scopes are to be searched for scope resolution.
a. LEGR **b. LEGB** c. LEBG d. LEGP

14. ____ scope is the lowest in hierarchy.
a. Local b. Enclosed c. Global **d. Built-in**

15. ____ scope is the highest in hierarchy.
a. Local b. Enclosed c. Global d. Built-in

16. There are____types of variable scope existing.
a. three b. two **c. four** d. five

17. ____ scope refers to variables defined in current function.
a. Local b. Enclosed c. Global d. Built-in

18. A function will first look up for a variable name in its _____ scope.
a. Local b. Enclosed c. Global d. Built-in

19. A variable which is declared outside of all the functions in a program is known as__variable.
a. Local b. Enclosed **c. Global** d. Built-in
20. ____ variable can be accessed inside or outside of all the functions in a program.
a. Local b. Enclosed **c. Global** d. Built-in
21. A function within another function is called ____ function.
a. recursive **b. nested** c. associated d. built-in
22. When a compiler or interpreter search for a variable in a program, it first search Local, and then search____scope.
a. Local **b. Enclosed** c. Global d. Built-in
23. The____scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter.
a. Local b. Enclosed c. Global **d. Built-in**
24. Any variable or module which is defined in the library functions of a programming language has____scope.
a. Local b. Enclosed c. Global **d. Built-in / Module**
25. The process of subdividing a computer program into separate sub-programs is called____programming.
a. Structured **b. Modular** c. Object Oriented d. Linear
26. ____ programming enables programmers to divide up the work and debug pieces of the program independently.
a. Structured **b. Modular** c. Object Oriented d. Linear
27. ____ is a module.
a. procedure b. subroutine c. function **d. All the above**
28. Modules contain ____
a. instructions b. processing logic
c. data **d. All the above**
29. ____ is a fundamental concept in security that minimizes risk to the object.
a. Access control b. Scope c. Module d. None of these
30. ____ is a selective restriction of access to data.
a. Access control b. Scope c. Module d. None of these
31. ____ members of a class are denied access from the outside the class.
a. public b. protected **c. private** d. perfect
32. ____ members are accessible from outside the class.
a. public b. protected c. private d. perfect
33. ____ members of a class are accessible from within the class and are also available to its sub-classes.
a. public **b. protected** c. private d. perfect
34. ____ enables specific resources of the parent class to be inherited by the child class.
a. public **b. protected** c. private d. perfect
35. Python prescribes a convention of prefixing the name of the variable or method with____underscore to emulate the behaviour of protected access specifiers.
a. single b. double c. no d. None of these
36. Python prescribes a convention of prefixing the name of the variable or method with____underscore to emulate the behaviour of private access specifiers.
a. single **b. double** c. no d. None of these
37. Any member can be accessed from outside the class environment in _____ language.
a. Python b. C++ c. Java d. All the above
38. Any member cannot be accessed from outside the class environment in ____ language
a. Python b. C++ **c. Java** d. Either B or C

39. ____ language control the access to class members by public, private and protected keywords.

a. Python b. C++ c. Java d. **b And c**

2-MARKS

ANSWER THE FOLLOWING QUESTIONS

1. What is a scope?

Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.

In other words, which parts of your program can see or use it.

2. Why scope should be used for variable? State the reason.

To limit a variable's scope to a single definition scope is needed. In this way, changes inside the function can't affect the variable on the outside of the function in unexpected ways.

3. What is mapping?

The process of binding a variable name with an object is called mapping. = (equal to sign) is used in programming languages to map the variable and object.

4. What do you mean by Namespaces?

Namespaces are containers for mapping names of variables to objects.

Names are mapped with objects (name: = object) in programming language.

This allows access to objects by names you choose to assign to them.

5. How Python represents the private and protected Access specifiers?

Python prescribes a convention of prefixing the name of the variable or method with single or double underscores to emulate the behaviour of protected and private access specifiers.

All members in a Python class are public by default.

6. What do you mean by LEGB rule?

The LEGB rule is used to decide the order in which the scopes are to be searched for scope resolution.

7. What is the output of the following pseudo code?

```
1. x:= 'outer x variable'
2. display():
3. x:= 'inner x variable'
4. print x
5. display()
```

OUTPUT:

outer x variable

inner x variable

8. Define : Modular programming

The process of subdividing a computer program into separate sub-programs is called Modular programming.

Modular programming enables programmers to divide up the work and debug pieces of the program independently.

9. Give example for modules.

The examples of modules are procedures, subroutines, and functions.

10. Define : Access control

Access control is a security technique that regulates who or what can view or use resources in a computing environment.

It is a fundamental concept in security that minimizes risk to the object.

3-MARKS

ANSWER THE FOLLOWING QUESTIONS

1. Define Local scope with an example.

Local scope refers to variables defined in current function. A function will first look up for a variable name in its local scope. Only if it does not find it there, the outer scopes are checked.

Example,

```
1. Disp():
2. a:=7
3. print a
4. Disp()
```

Entire program

```
Disp():
a:=7
print a
Disp ()
```

Output of the Program

7

On execution of the above code the variable a displays the value 7, because it is defined and available in the local scope.

2. Define Global scope with an example.

A variable which is declared outside of all the functions in a program is known as global variable.

The global variable can be accessed inside or outside of all the functions in a program.

Example,

```
1. a:=10
2. Disp():
3. a:=7
4. print a
5. Disp()
6. print a
```

Entire program

```
a:=10
Disp():
a:=7
print a
Disp 1 ():
print a
```

Output of the Program

7
10

On execution of the above code the variable a which is defined inside the function displays the value 7 for the function call Disp() and then it displays 10, because a is defined in global scope.

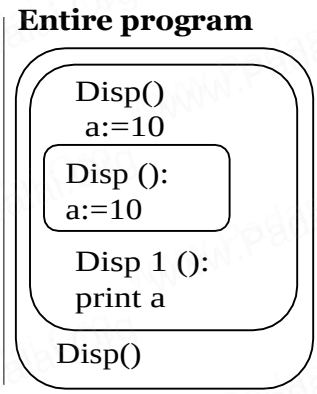
3. Define Enclosed scope with an example.

Enclosed Scope:

A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.

When a compiler or interpreter search for a variable in a program, it first search Local, and then search Enclosing scopes.

1. Disp():
2. a:=10
3. Disp():
4. print a
5. Disp 1()
6. print a
7. Disp()



Output of the Program
10
10

In the above example Disp1() is defined with in Disp(). The variable 'a' defined in Disp() can be even used by Disp1() because it is also a member of Disp().

4. Why access control is required?

Access control is a security technique that regulates who or what can view or use resources in a computing environment.

It is a fundamental concept in security that minimizes risk to the object. In other words access control is a selective restriction of access to data.

In object oriented programming languages it is implemented through access modifiers.

C++ and Java, control the access to class members by public, private and protected keywords.

Python prescribes a convention of prefixing the name of the variable or method with single or double underscore to emulate the behaviour of protected and private access specifiers.

5. Identify the scope of the variables in the following pseudo code and write its Output.

```
color:= Red
mycolor():
b:=Blue
myfavcolor():
g:=Green
print color, b, g
myfavcolor()
print color, b
mycolor()
print color
```

Scope of the variables:

Variable	Scope
color	Global
b	Enclosed
g	Local

Output:

Red Blue Green
Red Blue
Red

6. What do you mean by a module?

1. A module is a part of a program. Programs are composed of one or more independently developed modules.
2. A single module can contain one or several statements closely related each other.
3. Modules work perfectly on individual level and can be integrated with other modules.

7. How Python prescribe private and public access specifiers.

Python prescribes a convention of prefixing the name of the variable or method with single or double underscore to emulate the behaviour of protected and private access specifiers. All members in a Python class are public by default.

5-Marks

ANSWER THE FOLLOWING QUESTIONS

1. Explain the types of scopes for variable or LEGB rule with example.

The LEGB rule is used to decide the order in which the scopes are to be searched for scope resolution.

Local (L)	Define inside function/class
Enclosed(E)	Define inside enclosing functions(Nested function concept)
Global(G)	Defined at the uppermost level
Built-in(B)	Reserved names in built-in functions (modules)

Local Scope

Local scope refers to variables defined in current function. Always, a function will first look up for a variable name in its local scope. Only if it does not find it there, the outer scopes are checked.

Example:

```
1. Disp():
2. a:=7
3. print a
4. Disp()
```

Entire program

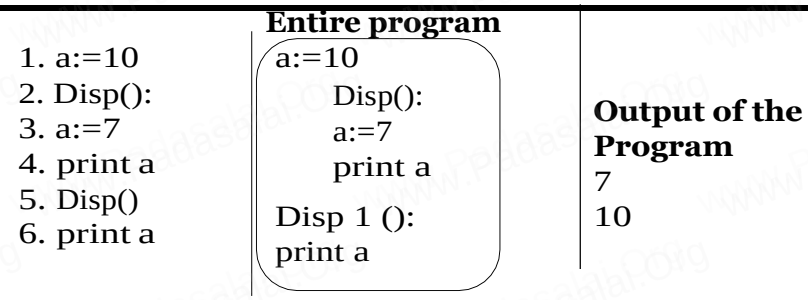
```
Disp():
a:=7
print a
Disp ()
```

Output of the Program

7

Global Scope

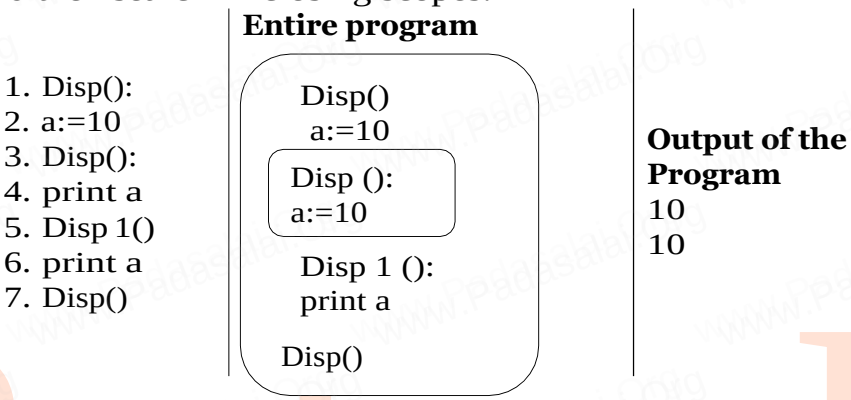
A variable which is declared outside of all the functions in a program is known as global variable. This means, global variable can be accessed inside or outside of all the functions in a program.



Enclosed Scope

Variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.

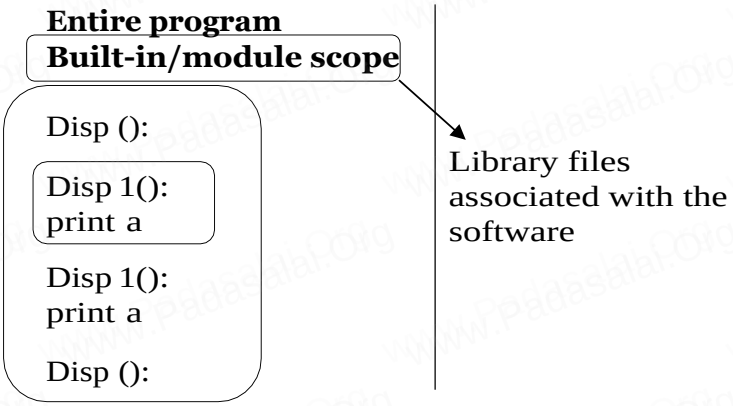
When a compiler or interpreter search for a variable in a program, it first search Local, and then search Enclosing scopes.



Built-in Scope

The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter. Any variable or module which is defined in the library functions of a programming language has Built-in or module scope.

They are loaded as soon as the library files are imported to the program.



2. Write any Five Characteristics of Modules.

Characteristics of Modules

- Modules contain instructions, processing logic, and data.
- Modules can be separately compiled and stored in a library.
- Modules can be included in a program.

- Module segments can be used by invoking a name and some parameters.
- Module segments can be used by other modules.

3. Write any five benefits in using modular programming.

The benefits of modular programming:

- Less code to be written.
- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand.
- Errors can easily be identified, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

HANDS ON PRACTICE

1. Observe the following diagram and Write the pseudo code for the following.

```
sum( ):
num1 : =  20
    sum1( ):
    num1 := num1 + 10
        sum2 ( )
        sum1 := num1 + 10
    sum2( )
sum 1()
num 1:=10
sum ()
print num 1
```

PSEUDO CODE:

```
1. sum( ):
2. num1 : = 20
3. sum1( ):
4. num1 := num1 + 10
5. sm2 ( )
6. num1 := num1 + 10
7. sum2( )
8. sum1( )
9. num1 := 10
10. sum( )
11. print num1
```

CHAPTER 4 ALGORITHMIC STRATEGIES

CHOOSE THE BEST ANSWER

1. The word comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-I Musa al Khowarizmi is called?
a) Flowchart b) Flow **c) Algorithm** d) Syntax

2. From the following sorting algorithms which algorithm needs the minimum number of swaps?
a) Bubble sort b) Quick sort c) Merge sort **d) Selection sort**

3. Two main measures for the efficiency of an algorithm are
a) Processor and memory b) Complexity and capacity
c) Time and space d) Data and space

4. The complexity of linear search algorithm is
a) O(n) b) O(log n) c) O(n2) d) O(n log n)

5. From the following sorting algorithms which has the lowest worst case complexity?
a) Bubble sort b) Quick sort **c) Merge sort** d) Selection sort

6. Which of the following is not a stable sorting algorithm?
a) Insertion sort **b) Selection sort** c) Bubble sort d) Merge sort

7. Time complexity of bubble sort in best case is
a) θ (n) b) θ (n log n) **c) θ (n2)** d) θ (n(log n)2)

8. The Θ notation in asymptotic evaluation represents
a) Base case **b) Average case** c) Worst case d) NULL case

9. If a problem can be broken into sub-problems which are reused several times, the problem possesses which property?
a) Overlapping sub-problems b) Optimal substructure
c) Memorization d) Greedy

10. In dynamic programming, the technique of storing the previously calculated values is called ?
a) Saving value property b) Storing value property
c) Memorization d) Mapping

11. A(n)___is a finite set of instructions to accomplish a particular task.
a. Algorithm b. Flow chart c. Walkthrough d. None of these

12. ___ is a step-by-step procedure for solving a given problem.
a. Algorithm b. Flow chart c. Walkthrough d. None of these

13. ___ are maintained and manipulated effectively through data structures.
a. Algorithm **b. Data** c. Program d. None of these

14. ___ is an example for data structure.
a. arrays b. structures / list
c. tuple / dictionary **d. All the above**

15. The way of defining an algorithm is called____strategy.
a. problem solvingb. solution **c. algorithmic** d. None of these

16. The word____has come to refer to a method to solve a problem.
a. Algorithm b. Flow chart c. Solution d. None of these

17. Identify the correct statement from the following:
a. Algorithms are generic and not limited to computer alone
b. An algorithm can be implemented in any suitable programming language.
c. Algorithm cannot be used in various real time activities.
d. Both a and b

18. A program can be implemented by____programming approach.
a. Structured b. Object Oriented**c. Both A and B** d. None of these

19. Efficiency of an algorithm is defined by the utilization of_____complexity.
a. Time and Space b. Time
c. Space d. None of these
20. ____ is a theoretical performance analysis of an algorithm.
a. Priori estimate b. Posteriori testing
c. Both A and B d. None of these
21. ____ is called performance measurement.
a. Priori estimate **b. Posteriori testing**
c. Both A and B d. None of these
22. In ____ analysis, actual statistics like running time and required for the algorithm executions are collected.
a. Priori estimate b. Posteriori testing
c. Both A and B d. None of these
23. ____ factor is measured by counting the number of key operations like comparisons in the sorting algorithm.
a. Time b. Space c. Both A and B d. None of these
24. ____ is measured by the maximum memory space required by the algorithm.
a. Time **b. Space** c. Both A and B d. None of these
25. Identify the correct statement from the following**:**
a. The efficiency of an algorithm is defined as the number of computational resources used by the algorithm.
b. A variable part is defined as the total space required by variables, which sizes depends on the problem and its iteration.
c. A fixed part is defined as the total space required to store certain data and variables for an algorithm.
d. All the above
26. The execution time of an algorithm depends on factor.
a. Speed of the machine b. Compiler and other system Software tools
c. Operating System **d. All the above**
27. The execution time of an algorithm depends on factor.
a. Programming language used b. Volume of data required
c. Speed of the machine **d. All the above d)**
28. ____ trade off refers to a situation where we can reduce the use of memory at the cost of slower program execution
a. Time b. Space c. Cost d. None of these
29. ____ trade off refers to a situation where we can reduce the running time at the cost of increased memory usage.
a. Time **b. Space** c. Cost d. None of these
30. The best algorithm to solve a given problem is one that ____
a. requires less space in memory b. takes less time to execute its instructions
c. Both A and B d. None of these
31. How many asymptotic notations are mostly used to represent time complexity of algorithms?
a. Five b. Four **c. Three** d. Two
32. ____ is an asymptotic notation used to represent time complexity of algorithm.
a. Big O b. Big Ω c. Big Θ **d. All the above**
33. ____ is often used to describe the worst-case of an algorithm.
a. Big O b. Big Ω c. Big Θ d. All the above
34. ____ is used to describe the best-case of an algorithm.
a. Big O **b. Big Ω** c. Big Θ d. All the above
35. ____ is used to describe lower bound of an asymptotic function.
a. Big O **b. Big Ω** c. Big Θ d. All the above
36. ____ is used to describe upper bound of an asymptotic function.

a. Big O b. Big Ω c. Big Θ d. All the above

37. Linear search also called____search.
a. Binary **b. Sequential** c. Random d. quick

38. ____ search is a sequential method for finding a particular value in a list.
a. Linear b. Sequential c. Binary **d. Either A or B**

39. In____searching algorithm, list need not be ordered.
a. Linear b. Sequential c. Binary **d. Either A or B**

40. ____ search also called half-interval search algorithm.
a. Linear b. Sequential **c. Binary** d. Either A or B

41. The ____ search algorithm can be done as divide-and-conquer search algorithm.
a. Linear b. Sequential **c. Binary** d. None of these

42. The____search algorithm executes in logarithmic time.
a. Binary b. Sequential c. Linear d. All of these

43. List of elements in an array must be sorted first for____search.
a. Linear b. Sequential **c. Binary** d. Unary

44. ____ sort is a simple sorting algorithm.
a. Bubble b. Merge c. Insertion d. Selection

45. ____ sort algorithm is too slow and less efficient when compared other sorting methods.
a. Bubble b. Merge c. Insertion d. Selection

46. ____ sort algorithm works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list.
a. Bubble b. Merge **c. Insertion** d. Selection

47. ____ algorithm uses n-1 number of passes to get the final sorted list.
a. Bubble b. Merge **c. Insertion** d. Selection

48. Dynamic programming approach is similar to ____
a. Divide and Conquer b. Integration
c. Merging d. None of these

49. ____ programming is used whenever problems can be divided into similar sub-problems.
a. Static **b. Dynamic** c. Concrete d. None of these

50. Dynamic algorithms uses____technique.
a. Memorization b. Memorize c. Memory d. None of these

51. ____ sort algorithm compares each pair of adjacent elements and swaps them if they are in the unsorted order.
a. Bubble b. Merge c. Insertion d. Selection

2 marks

ANSWER THE FOLLOWING QUESTIONS

1. What is an Algorithm?

An algorithm is a finite set of instructions to accomplish a particular task. It is a step-by-step procedure for solving a given problem.

2. Define Pseudo code.

It is an implementation of an algorithm in the form of annotations and informative text written in plain English.

It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

3. Who is an Algorist?

Algorist may refer to:

A person skilled in the technique of performing basic decimal arithmetic, known as algorism

One who practices algorism is known as an algorist.

A person skilled in the design of algorithms an algorithmic artist.

4. What is sorting?

Arranging the data in ascending or descending order is called sorting.

5. What is searching? Write its types.

Searching is the process of finding a particular data in a collection of data.

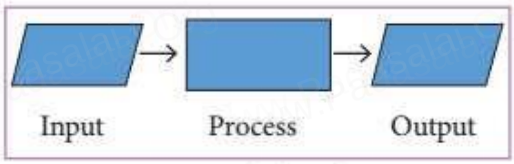
Types:

- Linear Search or Sequential Search
- Binary Search

6. What do you mean by algorithmic solution?

An algorithm that yields expected output for a valid input is called an algorithmic solution.

7. Draw picture to show the process of an algorithm.



A Typical Algorithm

8. What are the various data manipulations?

Data manipulations are Searching
Sorting
Inserting
Updating
Deleting an item.

9. What do you mean by algorithmic strategy?

The way of defining an algorithm is called algorithmic strategy.

10. Design an algorithm to find square of the given number and display the result.

Algorithm:

Step 1 – start the process
Step 2 – get the input x
Step 3 – calculate the square by multiplying the input value ie., square ← x* x
Step 4 - display the result square
Step 5 – stop

11. What do you mean by algorithmic solution?

An algorithm that yields expected output for a valid input is called an algorithmic solution.

12. Which is a best algorithm to solve a problem?

The best algorithm to solve a given problem is one that requires less space in memory and takes less time to execute its instructions to generate output.

13. Write note on binary search. Binary Search

Binary search also called half-interval search algorithm. It finds the position of a search element within a sorted array.

The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.

ANSWER THE FOLLOWING QUESTIONS (3 MARKS)

1. List the characteristics of an algorithm.

The characteristics of an algorithm:

- Input
- Output
- Finiteness
- Definiteness
- Effectiveness
- Correctness
- Simplicity
- Unambiguous
- Feasibility
- Portable
- Independent

2. Discuss about Algorithmic complexity and its types.

Computer resources are limited. Efficiency of an algorithm is defined by the utilization of time and space complexity.

Time Complexity: The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.

Space Complexity: Space complexity of an algorithm is the amount of memory required to run to its completion.

Example:

Suppose A is an algorithm and n is the size of input data, the time and space used by the algorithm A are the two main factors, which decide the efficiency of A.

Time Factor: Time is measured by counting the number of key operations like comparisons in the sorting algorithm.

Space Factor: Space is measured by the maximum memory space required by the algorithm. The complexity of an algorithm f(n) gives the running time and/or the storage space required by the algorithm in terms of n as the size of input data.

3. What are the factors that influence time and space complexity.

- The efficiency of an algorithm depends on how efficiently it uses time and memory space. They are depending on a number of factors such as:
- Speed of the machine Compiler and other system Software tools Operating System Programming language used Volume of data required

4. Write a note on Asymptotic notation. Asymptotic Notations

Asymptotic notations are languages that use meaningful statements about time and space complexity. The following three asymptotic notations are mostly used to represent time complexity of algorithms:

(i) Big O Big O is often used to describe the worst-case of an algorithm.

(ii) Big Ω Big Omega is the reverse Big O, if Big O is used to describe the upper bound (worst - case) of asymptotic function, Big Omega is used to describe the lower bound (best-case).

(iii) Big Θ When an algorithm has a complexity with lower bound = upper bound, say that an algorithm has a complexity O (n log n) and Ω (n log n), it's actually has the complexity Θ (n log n), which means the running time of that algorithm always falls in n log n in the best-case and worst-case.

5. What do you understand by Dynamic programming?

- ✓ Dynamic programming approach is similar to divide and conquer. The given problem is divided into smaller and yet smaller possible sub-problems.
- ✓ Dynamic programming is used whenever problems can be divided into similar sub-problems. so that their results can be re-used to complete the process.
- ✓ Dynamic programming approaches are used to find the solution in optimized way. For every inner sub-problem, dynamic algorithm will try to check the results of the previously solved sub-problems.
- ✓ The solutions of overlapped sub-problems are combined in order to get the better solution.

Steps to do Dynamic programming:

i. The given problem will be divided into smaller overlapping sub-problems.
ii. An optimum solution for the given problem can be achieved by using result of smaller sub-problem.
iii. Dynamic algorithms uses Memorization.

6. What are the advantages of Pseudocode? Advantages of Pseudocode.

Improves the readability of any approach.

Acts as a biridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudocode is written out.

The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

7. What are the phases of analysis of an algorithm?

Analysis of algorithms and performance evaluation can be divided into two different phases:

A Priori estimates: This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.

A Posteriori testing: This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.

8. Explain Space complexity. Space Complexity

Space complexity of an algorithm is the amount of memory required to run to its completion.

The space required by an algorithm is equal to the sum of the following two components:

• A fixed part is defined as the total space required to store certain data and variables for an algorithm. For example, simple variables and constants used in an algorithm.

• A variable part is defined as the total space required by variables, which sizes depends on the problem and its iteration. For example: recursion used to calculate factorial of a given value n.

9. Write about linear search. Linear Search

Linear search also called sequential search is a sequential method for finding a particular value in a list. This method checks the search element with each element in sequence until the desired element is found or the list is exhausted. In this searching algorithm, list need not be ordered.

ANSWER THE FOLLOWING QUESTIONS (5MARKS)

1. Explain the characteristics of an algorithm.

Input	Zero or more quantities to be supplied
Output	At least one quantity is produced
Finiteness	Algorithms must terminate after finite number of steps
Definiteness	All operations should be well defined. For example operations involving division by zero or taking square root for negative number are unacceptable
Effectiveness	Every instruction must be carried out effectively
Correctness	The algorithms should be error free
Simplicity	Easy to implement
Unambiguous	Algorithm should be clear and unambiguous. Each of its steps and their inputs/outputs should be clear and must lead to only one meaning
Feasibility	Should be feasible with the available resources.
Portable	An algorithm should be generic, independent of any programming language or an operating system able to handle all range of inputs.
Independent	An algorithm should have step-by-step directions, which should be independent of any programming code.

2. Discuss about Linear search algorithm. Linear Search

Linear search also called sequential search is an sequential method for finding a particular value in a list. This method checks the search element with each element in sequence until the desired element is found or the list is exhausted. In this searching algorithm, list need not be ordered.

Pseudo code

Traverse the array using for loop In every iteration, compare the target sea rch key value with the current value of the list.

1) If the values match, display the current index and value of the array.

2) If the values do not match, move on to the next array element.

If no match is found, display the search element not found. To search the number 25 in the array given below, linear search will go step by step in a sequential order starting from the first element in the given array if the search element is found that index is returned otherwise the search is continued till the last index of the array. In this example number 25 is found at index number 3.

Index	0	1	2	3	4
values	10	12	20	25	30

Example 1:
Input: values[ ] = {5, 34, 65, 12, 77, 35}
target = 77
Output: 4
Example 2:
Input: values[] = {101, 392, 1, 54, 32, 22, 90, 93}
target = 200
Output: -1 (Not Found)

3. What is Binary search? Discuss with example.

Binary Search

Binary search also called half-interval search algorithm. It finds the position of a search element within a sorted array. The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.

Pseudo code for Binary search

1. Start with the middle element:

If the search element is equal to the middle element of the array i.e., the middle value = number of elements in array/2, then return the index of the middle element.

If not, then compare the middle element with the search value, If the search element is greater than the number in the middle index, then select the elements to the right side of the middle index, and go to Step-1.

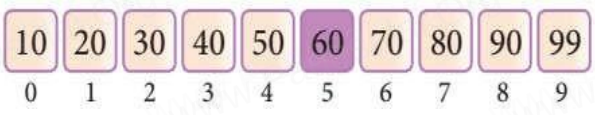
If the search element is less than the number in the middle index, then select the elements to the left side of the middle index, and start with Step-1.

2. When a match is found, display success message with the index of the element matched.

3. If no match is found for all comparisons, then display unsuccessful message.

Binary Search Working principles

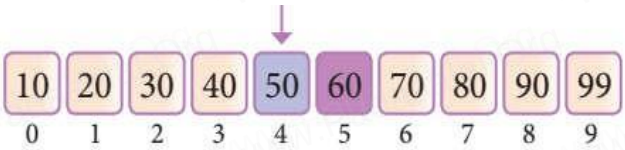
List of elements in an array must be sorted first for Binary search. The following example describes the step by step operation of binary search. Consider the following array of elements; the array is being sorted so it enables to do the binary search algorithm. Let us assume that the search element
is 60 and we need to search the location or index of search element 60 using binary search.



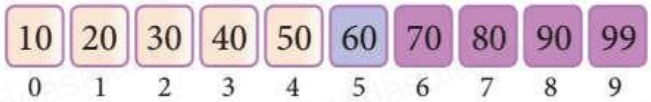
First, we find index of middle element of the array by using this formula :

mid = low + (high - low) / 2

Here it is, 0 + (9 - 0) / 2 = 4 (fractional part ignored). So, 4 is the mid value of the array.



Now compare the search element with the value stored at mid value location 4. The value stored at location or index 4 is 50, which is not match with
search element. As the search value 60 is greater than 50.

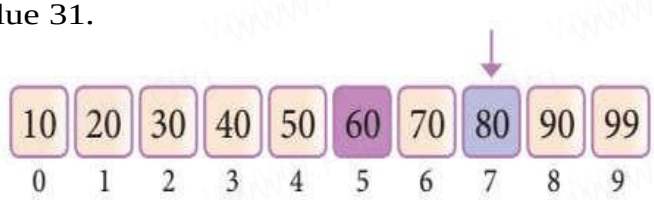


Now we change our low to mid + 1 and find the new mid value again using the formula.

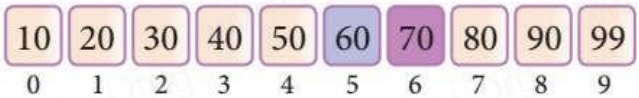
low to mid + 1

mid = low + (high - low) / 2

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.



The value stored at location or index 7 is not a match with search element, rather it is more than what we are looking for. So, the search element must be in the lower part from the current mid value location

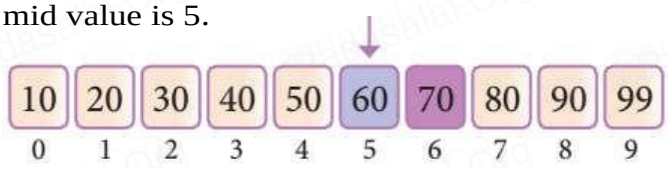


The search element still not found. Hence, we calculated the mid again by using the formula.

high = mid -1

mid = low + (high - low)/2

Now the mid value is 5.



Now we compare the value stored at location 5 with our search element. We found that it is a match. We can conclude that the search element 60 is found at location or index 5. For example if we take the search element as 95, For this value this binary search algorithm return unsuccessful result.

4. Explain the Bubble sort algorithm with example.

Bubble sort algorithm

Bubble sort is a simple sorting algorithm. The algorithm starts at the beginning of the list of values stored in an array. It compares each pair of adjacent elements and swaps them if they are in the unsorted order. This comparison and passed to be continued until no swaps are needed, which indicates that the list of values stored in an array is sorted.

The algorithm is a comparison sort, is named for the way smaller elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and less efficient when compared to insertion sort and other sorting methods.

Assume list is an array of n elements. The swap function swaps the values of the given array elements.

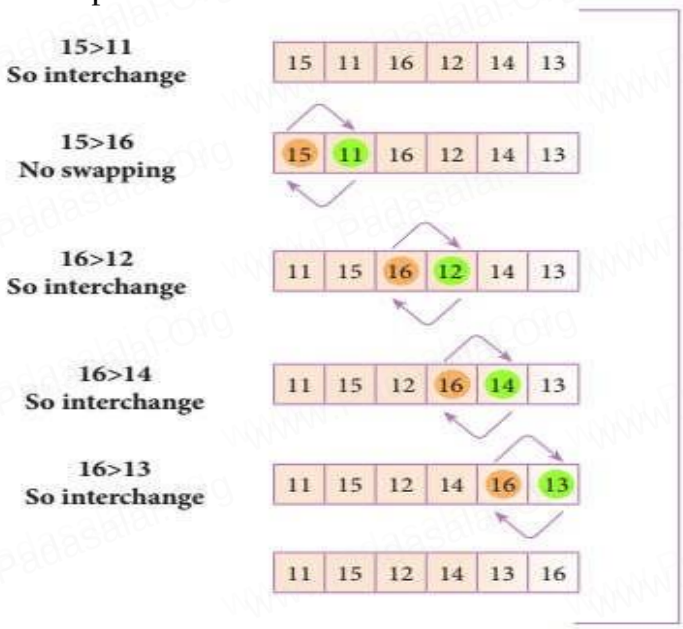
Pseudo code

Start with the first element i.e., index = 0, compare the current element with the next element of the array.

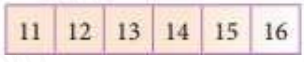
If the current element is greater than the next element of the array, swap them.

If the current element is less than the next or right side of the element, move to the next element. Go to Step 1 and repeat until end of the index is reached.

Let's consider an array with values {15, 11, 16, 12, 14, 13} Below, we have a pictorial representation of how bubble sort will sort the given array



The above pictorial example is for iteration-1. Similarly, remaining iteration can be done. The final iteration will give the sorted array. At the end of all the iterations we will get the sorted values in an array as given below



5. Explain the concept of Dynamic programming with suitable example.

Dynamic programming approach is similar to divide and conquer. The given problem is divided into smaller and yet smaller possible sub-problems.

Dynamic programming is used whenever problems can be divided into similar sub-problems. so that their results can be re-used to complete the process.

Dynamic programming approaches are used to find the solution in optimized way. For every inner sub-problem, dynamic algorithm will try to check the results of the previously solved sub-problems.

The solutions of overlapped sub-problems are combined in order to get the better solution.

Steps to do Dynamic programming:

The given problem will be divided into smaller overlapping sub-problems. An optimum solution for the given problem can be achieved by using result of smaller sub-problem.

Dynamic algorithms uses Memorization.

Example: Fibonacci Series generation

Fibonacci series generates the subsequent number by adding two previous numbers. Fibonacci series starts from two numbers - Fib0 & Fib1. The initial values of Fib0 & Fib1 can be taken as 0 and 1.

Fibonacci series satisfies the following conditions:

$Fib_n = Fib_{n-1} + Fib_{n-2}$

Hence, a Fibonacci series for the n value 8 can look like this

Fib_8 = 0 1 1 2 3 5 8 13

Fibonacci Iterative Algorithm with Dynamic programming approach

The following example shows a simple Dynamic programming approach for the generation of Fibonacci series.

Initialize f0=0, f1 =1

Step-1: Print the initial values of Fibonacci f0 and f1

Step-2: Calculate Fibonacci fib ← f0 + f1

Step-3: Assign f0← f1, f1← fib

Step-4: Print the next consecutive value of Fibonacci fib

Step-5: Goto step-2 and repeat until the specified number of terms generated For example if we generate Fibonacci series upto 10 digits, the algorithm will generate the series as shown below:

The Fibonacci series is : 0 1 1 2 3 5 8 13 21 34 55

6. Compare algorithm and a program.

Algorithm	Program
Algorithm helps to solve a given problem logically and it can be contrasted with the program.	Program is an expression of algorithm in a programming language.
Algorithm can be categorized	Algorithm can be implemented by

based on their implementation methods, design techniques	structured or object oriented programming approach
There is no specific rules for algorithm writing but some guidelines should be followed.	Program should be written for the selected language with specific syntax
Algorithm resembles a pseudo code which can be implemented in any language.	Program is more specific to a programming language

7. Explain Selection sort algorithm with example.

Selection sort

The selection sort is a simple sorting algorithm that improves on the performance of bubble sort by making only one exchange for every pass through the list. This algorithm will first find the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue until the entire array is sorted in respective order. This algorithm repeatedly selects the next-smallest element and swaps in into the right place for every pass. Hence it is called selection sort.

Pseudo code

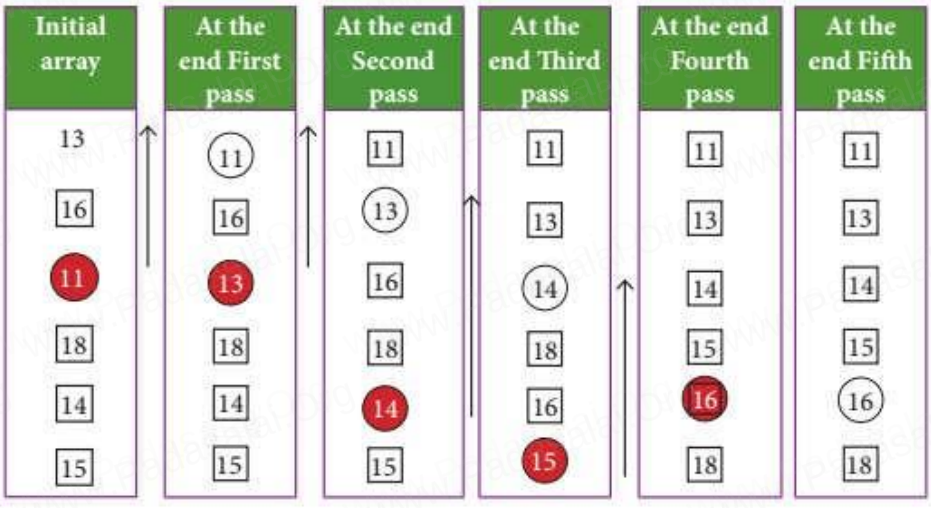
Start from the first element i.e., index-0, we search the smallest element in the array, and replace it with the element in the first position.

Now we move on to the second element position, and look for smallest element present in the sub-array, from starting index to till the last index of sub - array.

Now replace the second smallest identified in step-2 at the second position in the or original array, or also called first position in the sub array.

This is repeated, until the array is completely sorted.

Let's consider an array with values {13, 16, 11,18, 14, 15} Below, a pictorial representation of how selection sort will sort the given array is given:



SORTING PROCESS

In the first pass, the smallest element will be 11, so it will be placed at the first position.

After that, next smallest element will be searched from an array. Now we will get 13 as the smallest, so it will be then placed at the second position.

Then leaving the first element, next smallest element will be searched, from the remaining elements. We will get 13 as the smallest, so it will be then placed at the second position.

Then leaving 11 and 13 because they are at the correct position, we will search for the next smallest element from the rest of the elements and put it at third position and keep doing this until array is sorted.

Finally we will get the sorted array end of the pass as shown above diagram.

8. Explain Insertion sort algorithm with example.

Insertion sort is a simple sorting algorithm. It works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list. This algorithm builds the final sorted array at the end. This algorithm uses n-1 number of passes to get the final sorted list as per the pervious algorithm as we have discussed.

Pseudo for Insertion sort

Step 1 - If it is the first element, it is already sorted.
Step 2 - Pick next element
Step 3 - Compare with all elements in the sorted sub-list
Step 4 - Shift all the elements in the sorted sublist that is greater than the value to be sorted
Step 5 - Insert the value

44	16	83	07	67	21	34	45	10	Assume 44 is a soted list of 1 item
16	44	83	07	67	21	34	45	10	inserted 16
16	44	83	07	67	21	34	45	10	inserted 83
07	16	44	83	67	21	34	45	10	inserted 07
07	16	44	67	83	21	34	45	10	inserted 67
07	16	21	44	67	83	34	45	10	inserted 21
07	16	21	34	44	67	83	45	10	inserted 34
07	16	21	34	44	45	67	83	10	inserted 45
07	10	16	21	34	44	45	67	83	inserted 10

Step 6 - Repeat until list is sorted At the end of the pass the insertion sort algorithm gives the sorted output in ascending order as shown below:

07	10	16	21	34	44	45	67	83

algorithm will try to check the results of the previously solved sub-problems. The solutions of overlapped sub-problems are combined in order to get the better solution.

STUDY MATERIAL –PYTHON

Lesson [6-9]

Hands on Practice

12

Prepared By

Mr.M.Dhanapal,MCA.,B.Ed.,

PG-Assist In Computer Science

Literacy Mission MHSS, Samalapuram

Lesson 6

1. **Write a program to check whether the given character is a vowel or not.**

```
ch=input("enter the character:")
if ch in ('a','e','i','o','u','A','E','I','O','U'):
   print("Given character is Vowel")
else:
   print("Given Character is not Vowel")
```

Output:

```
enter the character:c
Given Character is not Vowel
>>>
```

Output:

```
enter the character:a
Given character is Vowel
>>>
```

2. **Using if..else..elif statement check smallest of three numbers.**

```
a=input("enter the Number 1:")
b=input("enter the Number 2:")
c=input("enter the Number 3:")
if a<b and a<c :
   print("The given  number {} is smallest".format(a))
elif b<a and b<c:
   print("The given  number {} is smallest".format(b))
else:
   print("The given  number {} is smallest".format(c))
```

Output:

```
enter the Number 1:5
enter the Number 2:4
enter the Number 3:2
The given  number 2 is smallest
>>>
```

3. **Write a program to check if a number is Positive, Negative or zero.**

```
a=int(input("enter the Number 1:"))
if a>0 :
   print("The given  number {} is Positive".format(a))
elif a<0:
   print("The given  number {} is Negative".format(a))
else:
   print("The given  number {} is Zero".format(a))
```

Output:

```
enter the Number 1:4
The given  number 4 is Positive
>>>
```

4. **Write a program to display Fibonacci series 0 1 1 2 3 4 5…… (upto n terms)**

```
a,b=0,1
n=int(input("Enter the number of terms to display fibbonaci:"))
print(a,end='\n')
print(b,end='\n')
for i in range(2,n,1):
   c=a+b
   print(c,end='\n')
   a=b
   b=c
```

Output:

```
Enter the number of terms to display fibbonaci:5
0
1
1
2
3
>>>
```

5. **Write a program to display sum of natural numbers, up to n.**

```
n=int(input("Enter the number of terms to find sun:"))
sum=0
for i in range(1,n+1):
  sum+=i
print("The sum of given number {} is {}".format(n,sum))
```

Output:

```
Enter the number of terms to find sun:10
The sum of given number 10 is 55
>>>
```

6. **Write a program to check if the given number is a palindrome or not.**

```
n=int(input("enter a number: "))
temp=n
r=0
while temp!=0:
   r=(r*10)+(temp%10)
   temp=temp//10
if n==r:
   print("number is palindrome")
else:
   print("number is not palindrome")
```

Output:

```
enter a number: 424
number is palindrome
>>>
```

7. **Write a program to print the following pattern**

```
* * * * *
* * * *
* * *
* *
*
for i in range(6,0,-1):
        j=1
        while(j<i):
                print("*",end=' ')
                j+=1
        print('\n')
```

Output:

```
* * * * *

* * * *

* * *

* *

*
```

8. . Write a program to check if the year is leap year or not.

```
y=int(input("Enter a year : "))
n=y%4
if (y==0):
print("The year {} is a Leap Year".format(y))
else:
print("The year {} is not a Leap Year".format(y))
```

Output:

```
Enter a year : 2001
The year 2001 is not a Leap Year
>>>
```

Lesson 7

1. **Try the following code in the program**

```
def printinfo( name, salary = 3500):
    print (“Name: “, name)
    print (“Salary: “, salary)
    return
printinfo(“Mani”)
```

Slno	code	Result
1	printinfo(“3500”)	Name:3500 Salary:3500
2	printinfo(“3500”,”Sri”)	Name:3500 Salary:Sri
3	printinfo(name=”balu”)	Name:balu Salary:3500
4	printinfo(“Jose”,1234)	Name:jose Salary:1234
5	printinfo(“ ”,salary=1234)	Name: Salary:1234

2. **Evaluate the following functions and write the output**

Slno	Function	Output
1	eval(‘25*2-5*4')	30
2	math.sqrt(abs(-81))	9.0
3	math.ceil(3.5+4.6)	9
4	math.floor(3.5+4.6)	8

3. **Evaluate the following functions and write the output**

Slno	function	Output
1	1) abs(-25+12.0)) 2) abs(-3.2)	13.0 3.2
2	1) ord('2') 2) ord('$')	50 36
3	type('s')	<class 'str'>
4	bin(16)	0b10000
5	1) chr(13) 2) print(chr(13))	'\r'
6	1) round(18.2,1) 2) round(18.2,0) 3) round(0.5100,3) 4) round(0.5120,3)	18.2 18.0 0.51 0.512
7	1) format(66, 'c') 2) format(10, 'x') 3) format(10, 'X') 4) format(0b110, 'd') 5) format(0xa, 'd')	'B' 'a' 'A' '6' '10'
8	1) pow(2,-3) 2) pow(2,3.0) 3) pow(2,0) 4) pow((1+2),2) 5) pow(-3,2) 6) pow(2*2,2)	0.125 8.0 1 9 9 16

Lesson 8

1. Write a python program to find the length of a string.

```
a=input("Enter the string:")
x=len(a)
print("The Length of the given string :",x)
```

Output:

```
Enter the string:welcome to literacy
The Length of the given string : 19
>>>
```

2.Write a program to count the occurrence of each word in a given

```
a=input("Enter the string:")
#b=input("enter the word to count in the string:")
for i in a:
  print("the character is ",i,"is ",a.count(i)," times")
```

Output:

```
Enter the string:welcome
the character is  w is  1  times
the character is  e is  2  times
the character is  l is  1  times
the character is  c is  1  times
the character is  o is  1  times
the character is  m is  1  times
the character is  e is  2  times
```

3.Write a program to add a prefix text to all the line in a string.

```
b="python is "
a=input("Enter the string:")
b+=a
print(b)
```

Output:

```
Enter the string:programming language
python is programming language
>>>
```

4. Write a program to print integers with '*' on the right of specified width.

```
y=int(input("enter the number: "))
z=int(input("how many * u need right side-specify the width: "))
x='*'
print(y,x*z)
```

Output:

```
enter the number: 123
how many * u need right side-specify the width: 2
123 **
>>>
```

5. Write a program to create a mirror of the a mirror of the given string

```
    str1 = input("Enter a string:  ")
    c=""
    index = -1
```

```
for i in str1:
    c += str1[index]
    index-=1
print("The mirror of the word is : ",c)
```

OUTPUT:

```
Enter a string:  Well
The mirror of the word is :  lleW
>>>
```

6. Write a to remove all the occurrences of a given character in a string.

```
str2=input("Enter a string: ")
str1=input("Enter a charactoe to remove:  ")
f=""
for i in str2:
        if(str1==i):
                pass
        else:
                f+=i
print("The string without {} is {} ".format(str1,f))
```

OUTPUT:

```
Enter a string: ate
Enter a charactoe to remove:  e
The string without e is at
>>>
```

7. Write a program to append a string to another string without using += operator.

```
a=input("Enter the string1 :")
b=input("Enter the string2 :")
c=a+b
print("the combined string is :", c)
```

Output:

Enter the string1 :welcome

Enter the string2 :to python

the combined string is : welcome to python

8.Write a program to Swap a string.

```
str1 = input("Enter a stering 1")
str2 = input("Enter a stering 2")
cha = str1
str1 = str2
str2=cha
print("The swap of String 1 is: ",str1)
print("The swap of String 2 is: ",str2)
```

OUTPUT:

Enter a stering 1Literacy

Enter a stering 2School

The swap of String 1 is: School

The swap of String 2 is: Literacy

9.Write a program to replace a string with another string without using replace().

```
s= input("Enter a string: ")
print("String before replaced: ",s)
s= input("Enter a string to replace:  ")
print("String after replacement: ",s)
```

OUTPUT:

```
Enter a string: How are You ?
String before replaced:  How are You ?
Enter a string to replace: Iam Fine.
String after replacement:  Iam Fine.
```

10. Write a program to count the number of characters, words and lines in a given string

```
s=input("Enter string: ")
char=0
word=1
for i in s:
    char=char+1
    if(i==' '):
        word=word+1
print("Number of words in the string: ",word)
print("Number of characters in the string: ",char)
```

Output:

```
Enter string: literacy mission
Number of words in the string:  2
Number of characters in the string:  16
```

Lesson 9

1. **Write a program to remove duplicates from a list.**

```
def Remove(d):
    final_list = []
    for num in d:
        if num not in final_list:
            final_list.append(num)
    return final_list
d = [2, 4, 10, 20, 5, 2, 20, 4]
print("Original List: ",d)
print("After removing duplicate in list ",Remove(d))
```

Output:

```
Original List:  [2, 4, 10, 20, 5, 2, 20, 4]
After removing duplicate in list  [2, 4, 10, 20, 5]
>>>
```

2. **Write a program that prints the maximum value in a Tuple.**

```
tup=(56,34,58,12,35,98)
print("The maximum value in tupil: ",max(tup))
```

Output:

```
The maximum value in tupil:  98
>>>
```

3. Write a program that finds the sum of all the numbers in a Tuples using while loop.

```
tup=(1,6,5,3,4,9)
i=0
sum=0
a=len(tup)
while i<a :
   sum=sum+tup[i]
   i+=1
print(" the sum of the tupil :",sum)
```

Output:

```
the sum of the tupil : 28
```

4. Write a program that finds sum of all even numbers in a list.

```
tup=(1,6,5,3,4,9)
i=0
sum=0
a=len(tup)
while i<a :
   if tup[i]%2==0:
      sum=sum+tup[i]
   i+=1
print(" the sum of the even number in this tupil :",sum)
```

Output:

```
the sum of the even number in this tupil :10
```

5. Write programs that reverse a list using a loop.

```
def rev(a):
    b=[]
    i=-1
    while i>=-5:
        b.append(a[i])
        i=i+-1
    print("after reverse: ",b)
a=[1,2,3,4,5]
print("Before reverse: ",a)
rev(a)
```

Output:

```
Before reverse:  [1, 2, 3, 4, 5]
after reverse:  [5, 4, 3, 2, 1]
>>>
```

6. Write a program to insert a value in a list at the specified location.

```
a=[1,2,3,4,5]
print(a)
b=int(input("enter the specified location to insert  "))
c=int(input("enter the value to insert: "))
a.insert(b,c)
print("The updated list :",a)
```

Output:

[1, 2, 3, 4, 5]

enter the specified location to insert 3

enter the value to insert: 6

The updated list : [1, 2, 3, 6, 4, 5]

>>>

7. Write a program that creates a list of numbers from 1 to 50 that are either divisible by 3 or divisible by 6.

```
a=[]
for i in range(1,51):
  if i%3==0 or i%6==0:
    a.append(i)
print("These are the number divisible by 3 or 6 \n",a)
```

Output:

These are the number divisible by 3 or 6

[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48]

>>>

8. Write a program to create a list of numbers in the range 1 to 20. Then delete all the numbers from the list that are divisible by 3.

```
a=[]
for i in range(1,21):
   a.append(i)
print(a)
```

```
for i in a:
    if i%3==0:
        a.remove(i)
print("These are the number not divisible by 3 \n",a)
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
These are the number not divisible by 3
 [1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20]
>>>
```

9. **Write a program that counts the number of times a value appears in the list. Use a loop to do the same.**

```
list=[4,6,8,12]
for i in list:
    print(" The number",i," is ",list.count(i)," times")
```

Output:

```
The number 4  is  1  times
 The number 6  is  1  times
 The number 8  is  1  times
 The number 12  is  1  times
>>>
```

10. Write a program that prints the maximum and minimum value in a dictionary.

```
a={1:45,2:58,3:66,4:12,5:4}

print(a)

b=max(a, key = lambda x: a.get(x) )

c=min(a, key = lambda x: a.get(x) )

print("the maximum in the given dictionary ",a[b])

print("the minimum in the given dictionary ",a[c])
```

Output:

```
{1: 45, 2: 58, 3: 66, 4: 12, 5: 4}

the maximum in the given dictionary  66

the minimum in the given dictionary  4

>>>
```

"All the Best"

COMPUTER SCIENCE

PYTHON MANUAL LESSON 6-9

BOOK BACK PROGRAMS

NAME :

CLASS :

SCHOOL :

Prepared By

Mr.M.Dhanapal,MCA.,B.Ed.,

PG-Assist In Computer Science

Literacy Mission MHSS, Samalapuram

PYTHON - Lesson 6-9 Book back Programs | 12

1. **Write a program to display all 3 digit odd number.**

```
for i in range(100,1000,1):
    if i%2!=0:
        print(i,end=" ")
```

Output :

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python37-32/ex.py ==
101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141
143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183
185 187 189 191 193 195 197 199 201 203 205 207 209 211 213 215 217 219 221 223 225
227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267
269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309
311 313 315 317 319 321 323 325 327 329 331 333 335 337 339 341 343 345 347 349 351
353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387 389 391 393
395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435
437 439 441 443 445 447 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477
479 481 483 485 487 489 491 493 495 497 499 501 503 505 507 509 511 513 515 517 519
521 523 525 527 529 531 533 535 537 539 541 543 545 547 549 551 553 555 557 559 561
563 565 567 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603
605 607 609 611 613 615 617 619 621 623 625 627 629 631 633 635 637 639 641 643 645
647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687
689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729
731 733 735 737 739 741 743 745 747 749 751 753 755 757 759 761 763 765 767 769 771
773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807 809 811 813
815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855
857 859 861 863 865 867 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897
899 901 903 905 907 909 911 913 915 917 919 921 923 925 927 929 931 933 935 937 939
941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981
983 985 987 989 991 993 995 997 999
>>>
```

2. **Write a program to display multiplication table for a given number.**

```
a=int(input("Enter the number for multiplication table:"))
i=1
while(i<=10):
    z=i*a
    print (" %dX%d=%d"%(i,a,z))
    i=i+1
```

Output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python37-32/ex2.py ==
Enter the number for multiplication table:4
 1X4=4
 2X4=8
 3X4=12
 4X4=16
 5X4=20
 6X4=24
 7X4=28
 8X4=32
 9X4=36
 10X4=40
>>>
```

3. **Using if..else..elif statement write a suitable program to display largest of 3 numbers.**

```
a=int(input("\nEnter the number1:"))
b=int(input("\nEnter the number1:"))
c=int(input("\nEnter the number1:"))
if a>b and a>c:
   print("\n The given number %d is Greater"%(a))
elif b>a and b>c :
   print("\n The given number %d is Greater"%(b))
else:
   print("\n The given number %d is Greater"%(c))
```

Output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python37-32/ex2.py ==

Enter the number1:456

Enter the number1:856

Enter the number1:420

 The given number 856 is Greater
>>>
```

4. Write a program to display

```
A
A       B
A       B       C
A       B       C       D
A       B       C       D       E
```

```
i=65
while(i<=70):
    for j in range(65,i):
        print(chr(j),end="\t")
    print(end='\n')
    i=i+1
```

OUTPUT

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python37-32/ex4.py ==

A
A       B
A       B       C
A       B       C       D
A       B       C       D       E
>>>
```

5. **Write a Python program to display the given pattern**

```
COMPUTER
COMPUTE
COMPUT
COMPU
COMP
COM
CO
C
```

```
str="COMPUTER"
index=9
for i in str:
    print(str[:index-1])
    index-=1
```

OUTPUT:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python37-32/ex.py ==
COMPUTER
COMPUTE
COMPUT
COMPU
COMP
COM
CO
C
>>>
```

6. **Write a python code to check whether a given year is leap year or not.**

```
y=int(input("Enter a year :  "))
n=y%4
if (y==0):
   print("The year {} is a Leap Year".format(y))
else:
   print("The year {} is not a Leap Year".format(y))
```

Output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: \\S34\Users\Public\les 7.py ==========
Enter a year :  2001
The year 2001 is not a Leap Year
>>>
```

7. **Write a python code to find the L.C.M of two numbers.**

```
a=int(input("Enter the first number:"))
b=int(input("Enter the second number:"))
if(a>b):
   min1=a
else:
```

```
    min1=b
while(1):
    if(min1%a==0 and min1%b==0):
        print("LCM is:",min1)
        break
    min1=min1+1
```

Output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/DHANA RANJI/AppData/Local/Programs/Python/Python37-32/LCM.PY
Enter the first number:4
Enter the second number:2
LCM is: 4
>>>
```

Book Back – Output Programs

1. What will be the output of the following python code?

```
str1 = “School”
print(str1*3)
```

Output:

SchoolSchoolSchool

2. What will be the output of the given python program?

```
str1 = "welcome"
str2 = "to school"
str3=str1[:2]+str2[len(str2)-2:]
```

Output:

Weol

3. What will be the value of x in following python code?

```
List1=[2,4,6[1,3,5]]
    x=len(List1)
```

Value of X is **4**

4. What will be the output of the following code?

```
list = [2**x for x in range(5)]
print(list)
```

Output:

[1, 2, 4, 8, 16]

Computer Science - One mark Test (lesson 9-12

Literacy Mission Matric Hr Sec School, Samalapuram

Name : **Class :** **Date :** **Marks:50**

I. Choose the best answer

1. Python assigns an automatic index value for each of a list begin with _____.
   a) 0 b) -1 c) 1 d) none
2. Find the output of the following snippet
   >>Marks=[10,23,41,75]
   >> print(Marks[-3]
   a) 10 b) 23 c) 41 d) 75
3. ____ function in python is used to find the length of the list.
   a) List b) cpy() c) len() d) print()
4. A list element or range of elements can be changed or altered by using _____
   a) = b) + c)* d) -
5. _____ function in python is used to add more elements.
   a) append() b) insert() c) len() d) none of these
6. how many ways to delete an element from a list?
   a) 1 b) 2 c) 3 d) 4
7. _____ function is used to delete all elements in list.
   a) Remove() b) del() c) clear() d) pop()
8. _____ function is used to generate a series of values .
   a) Range() b) sum() c) pop() d) clear()
9. for x in range(3,10,3)
   print(x)
   What is the output?
   a) 3 6 9 1 b) 3 4 5 6 7 8 9 10 c) error d) 3 6 9
10. ___ function returns the number of similar elements present in the list.
    a) Count() b) reverse() c) sort() d) copy()
11. ____ is created by placing all the elements separated by comma within {}.
    a) List b) Tuples c) Set d) all the above
12. A={1,3,5,7}
    B={3,5,9,11}
    print(A-B)
    a) 1 7 b) 1 3 5 9 7 11 c) 5 9 11 d) 3 5
13. In python keys are separeted using ___ in dictionary.
    a) : b) , c) ? d) –

Computer Science - One mark Test (lesson 9-12

14.___ is a mixed collection of elements.
a) List b) Dictionary c) sets d) tuples

15._____ is called as sequence data type.
a) List b) Dictionary c) sets d) tuples

16. Every class has a unique name followed by _____
a) : b) ; c) . d) @

17._____ is a special function that is automatically executed when an object of a class is created.
a) Destructor b) Constructor c) main d) array

18.The variables defined inside the class is ____ by default.
a) Public b) private c) Protected d) local

19.A variable prefixed with double underscore becomes in _____ nature.
a) private b) protected c) public d) global

20. The process of creating objects is called as _____
a) Class b) class installation c) constructor d) object

21.Which of the following is automatically executed when class object created?
a) __object__() b) __del__() c) __func()__ d) __init__()

22.Class members are accessed through___.
a) Dot b) Comma c) colon d) []

Read the following program and answer the Questions :23-26

```
class Sample:
    def __init__(self, num):
        self.num=num
        print(num)
S=Sample(10)
```

23.What is the name of the class ?
a) Sample b) Student c) self d) init

24.Which one is the class member of sample?
a) Sample b) s c) S d) num

25.Which one of the following is object of class sample?
a) Sample b) s c) S d) num

26.What is the output of the above program?
a) 20 b) 15 c) 10 d) 5

27._____ is a repository of collection of related data organized in structure.
a) Data b) Database c) Information d) Field

Computer Science - One mark Test (lesson 9-12

28._____ is the entire collection of related data in one table.
a) Table b) File c) A and B d) none

29._____ model was developed by IBM as IMS.
a) Relational b) Hierarchy c) Network d) object

30.Diamond shape represent the _____
a) Entity b) attributes c) relation d) flow

31.Relational algebra was first created by ____
a) Edgar F codd b) chen c) a and b d) none

32.Data normalization was proposed by ____.
a) Edgar F codd b) chen c) a and b d) none

33.ER model was developed by
a) Edgar F codd b) chen c) a and b d) none

34.A tuple is also called as
a) Table b) row c) attribute d) field

35.SQL expansion _____.
a) Structured query language b) structured queue language
c) Senior queue language d) structured query list

36.Which language is used for defining the structure of the database.
a) DDL b) DML c) DQL d) TCl

37.DML has classied into ____
a) 2 b) 3 c) 4 d) 5

38.____ is used to give privellages and grants ,revoke
a) DDL b) DML c) DCL d) DQL

39._____ is condition applicable on a field or set of fields.
a) Query b) Field c) constraint d) key

40._____ is used for apply to a group of one or more columns.
a) Table b) Filed c) table constraint d) none

41.____ command is used to alter the structure of the table.
a) Alter b) drop c) Create d) Delete

42._____ keyword is used to eliminate the duplicate rows in table.
a) All b) distinct c) select d) none

43.which keyword is used in order by clause to display the data in descending order?
a) asce b) desc c) sort d) reverse

44._____ command is used to save the transaction permanently save any transaction.
a) rollback b) save c) commit d) create

Computer Science - One mark Test (lesson 9-12

45._____ command is used to generate queries in dbms.
a) SQL b) TCL c) DQL d) DML

46.Queries can be generated using _____
a) select b) drop c) create d) alter

47._____ keyword is used to specify a value which must be matched.
a) in b) not in c) between d) not between

48._____ command is used to update the data in the table.
a) alter b) update c) set d) where

49._____ may use relational and logical operations for condition.
a) check b) update c) where d) in

50._____ constraints ensures that no two rows have the same value in specific column.
a) null b) unique c) table d) constant

All the Best