SHRI KRISHNA ACADEMY

NEET,JEE & BOARD EXAM(10th,+1,+2) COACHING CENTRE
SBM SCHOOL CAMPUS, TRICHY MAIN ROAD,NAMAKKAL
CELL: 99655 31727 , 94432 31727

XII – COMPUTER SCIENCE

MATERIAL

2019 – 2020

DEPARTMENT OF COMPUTER SCIENCE

XII COMPUTER SCIENCE

UNIT – 1 PROBLEM SOLVING TECHNIQUES

CHAPTER-1 FUNCTIONS

Part - I (1 Mark)

Choose the best answer:

1. The small sections of code that are used to perform a particular task is called
   (A) Subroutines (B) Files (C) Pseudo code (D) Modules
2. Which of the following is a unit of code that is often defined within a greater code structure?
   (A) Subroutines **(B) Function** (C) Files (D) Modules
3. Which of the following is a distinct syntactic block?
   (A) Subroutines (B) Function **(C) Definition** (D) Modules
4. The variables in a function definition are called as
   (A) Subroutines (B) Function (C) Definition **(D) Parameters**
5. The values which are passed to a function definition are called
   (A) Arguments (B) Subroutines (C) Function (D) Definition
6. Which of the following are mandatory to write the type annotations in the function definition?
   (A) Curly braces **(B) Parentheses** (C) Square brackets (D) indentations
7. Which of the following defines what an object can do?
   (A) Operating System (B) Compiler **(C) Interface** (D) Interpreter
8. Which of the following carries out the instructions defined in the interface?
   (A) Operating System (B) Compiler **(C) Implementation** (D) Interpreter
9. The functions which will give exact result when same arguments are passed are called
   (A) Impure functions (B) Partial Functions
   (C) Dynamic Functions **(D) Pure functions**
10. The functions which cause side effects to the arguments passed are called
    (A) impure function (B) Partial Functions
    (C) Dynamic Functions (D) Pure functions

Part - II (2 Marks)

Answer the following questions

1. **What is a 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.
   - In Programming languages these subroutines are called as Functions.
2. **Define Function with respect to Programming language.**
   - A function is a unit of code that is oft en 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.

3. **Write the inference you get from X: = (78).**
   - X:=(78) has an expression in it but (78) is not itself an expression. Rather, it is a function definition. Definitions bind values to names, in this case the value 78 being bound to the name 'X'.
   - Definitions are not expressions, at the same time expressions are also not treated as definitions. Definitions are distinct syntactic blocks.
   - Definitions can have expressions nested inside them, and vice-versa.
4. **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

5. **Which of the following is a normal function definition and which is recursive function definition**

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
(iii) Recursive function definition

Part - III (3 Marks)

Answer the following questions:

1. **Mention the characteristics of Interface.**
   - The class template specifies the interfaces to enable an object to be created and operated properly.
   - An object's attributes and behaviour is controlled by sending functions to the object.
2. **Why strlen is called pure function?**

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

3. What is the side effect 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.
- When a function depends on variables or functions outside of its definition 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.

Example:

```
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.

4. Differentiate pure 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().
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 may modify the arguments which are passed to them

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

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 definition due to which the result will change each time. The side effect 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 effect) is that it doesn't take any arguments and it doesn't return any value.

Part - III (5Marks)

Answer the following questions:

1. 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.

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.
- The 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.

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.

2. 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

Ans:

- gcd
- let rec gcd
- a , b
- gcd b(a mod b)
- return a

3. 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 behavior 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.

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

4. Explain with an example interface and 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.

- class declaration combines the external interface (its local state) with an implementation of (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.

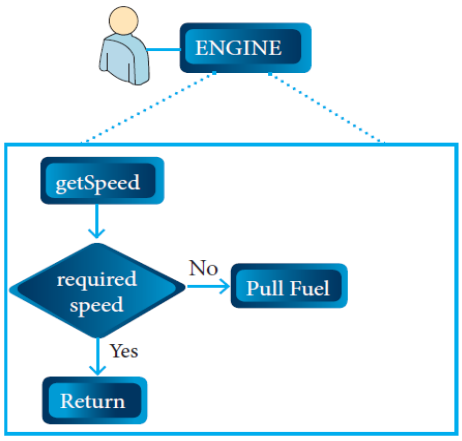
The difference between interface and implementation is

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

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

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. For example, let's take the example of increasing a car's speed.



- 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
```

INTERIOR ONE MARK:

1. ---------- are the basic building blocks of compute r programs.
   (a) Files **(b) Subroutines** (c) Code (d) Modules
2. In programming languages subroutines are called as ----------
   (a) Subroutines **(b) functions** (c) modules (d) files
3. Choose the correct Pair:
   a) smaller code structure - function
   b) functions - Subroutines
   c) Set of action - implementation
   d) Class - only one instance
4. ------- bind values to names
   (a) functions **(b) definitions** (c) modules (d) files
5. A function definition which call itself is called ----- function
   (a) return **(b) recursive** (c) module (d) definition
6. All functions are ------- definitions
   (a) static (b) dynamic (c) recursive (d) function
7. An ------- is an instance created from the class
   (a) class **(b) object** (c) code (d) interface
8. An object is also called ---------
   (a) instance (b) class (c) code (d) instruction
9. The ------ defines an objects visibility to the outside world
   (a) class **(b) interface** (c) implementation (d) program
10. Assertion : An interface is a set of action that an object can do.
    Reason : Interface defines an object's visibility to the outside world.
    a) Assertion is correct and reason is wrong **b) both are correct**
    c) Assertion is wrong and reason is correct d) both are wrong
11. How the object is processed and executed is the ---------
    (a) interface **(b) implementation** (c) class (d) program
12. Evaluation of ---------- functions does not cause any side effects to its output.
    (a) impure **(b) pure** (c) interface (d) class
13. -------- functions have side effects
    (a) impure (b) pure (c) interface (d) object
14. Pick out odd one from the following
    (a) class (b) object (c) interface **(d) file**
15. Match the following
    (a) rec - same arguments
    (b) (a:int) - recursive function

(c) let - parameter with type
(d) pure function - keyword

(a) 2 3 4 1 (b) 3 2 1 4
(c) 4 3 2 1 (d) 2 4 3 1

INTERIOR 2 MARKS:

1. **Define Parameter and arguments.**
   Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.
2. **Write the syntax for function definition**
   let rec fna1 a2 ... an := k
3. **What is recursive function**
   A function definition which calls itself is called recursive function.
4. **Define interface.**

❖ The interface defines an object's visibility to the outside world.
❖ Interface just defines what an object can do, but won't actually do it

5. **Define implementation**
   Implementation carries out the instructions defined in the interface

INTERIOR 3 MARKS:

1. **Write note on pure function with example.**
   Pure functions are functions which will give exact result when the same arguments are passed.
   Example: **let square x**
   return: x * x
2. **Write about impure function.**
   The variables used inside the function may cause side effects though the functions which are not passed with any arguments.
   Example:
```
let Random number
let a := random()
    if a > 10 then
        return: a
    else
        return: 10
```

INTERIOR 5 MARKS:

1. Problems.

CHAPTER -2 DATA ABSTRACTION

Part - I (1 Mark)

Choose the best answer:

1. Which of the following functions that build the abstract data type ?
   (A) Constructors (B) Destructors (C) recursive (D)Nested
2. Which of the following functions that retrieve information from the data type?
   (A) Constructors **(B) Selectors** (C) recursive (D)Nested

3. The data structure which is a mutable ordered sequence of elements is called
   (A) Built in **(B) List** (C) Tuple (D) Derived data
4. A sequence of immutable objects is called
   (A) Built in (B) List **(C) Tuple** (D) Derived data
5. The data type whose representation is known are called
   (A) Built in datatype (B) Derived datatype
   (C) Concrete datatype (D) Abstract datatype
6. The data type whose representation is unknown are called
   (A) Built in datatype (B) Derived datatype
   (C) Concrete datatype **(D) Abstract datatype**
7. Which of the following is a compound structure?
   (A) Pair (B) Triplet (C) single (D) quadrate
8. Bundling two values together into one can be considered as
   (A) Pair (B) Triplet (C) single (D) quadrat
9. Which of the following allow to name the various parts of a multi-item object?
   (A) Tuples (B) Lists **(C) Classes** (D) quadrat
10. Which of the following is constructed by placing expressions within square brackets?
    (A) Tuples **(B) Lists** (C) Classes (D) quadrats

Part - II (2 Marks)

Answer the following questions:

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

2. Differentiate constructors and selectors.

Constructor	Selector
Constructors are functions that build the abstract data type.	Selectors are functions that retrieve information from the data type.
Constructors create an object, bundling together different pieces of information	Selectors extract individual pieces of information from the object.
Example: city = makecity (name, lat, lon)	**Example:** getname(city) getlat(city) getlon(city)

3. **What is a Pair? Give an example.**
   - Python provides a compound structure called Pair which is made up of list or Tuple.
   - 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.
   - **Example:** lst:=[10,20]
4. **What is a List? Give an 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].
5. **What is a Tuple? Give an example.**
   - A tuple is a comma-separated sequence of values surrounded with parentheses. Tuple is similar to a list.
   - 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.
   - **Example:** colour= ('red', 'blue', 'Green')

Part - III (3 Marks)

Answer the following questions:

1. **Differentiate Concrete data type and abstract data type.**

Concrete Data type	Abstract Data type
Concrete data types or structures (CDT's) are direct implementations of a relatively simple concept.	Abstract Data Types (ADT's) offer a high level view (and use) of a concept independent of its implementation.
A concrete data type is a data type whose representation is known.	In abstract data type the representation of a data type is unknown.

2. **Which strategy is used for program designing? Define that Strategy.**
   - **Wishful Thinking** strategy is used for program designing.
   - Wishful Thinking is the formation of beliefs and making decisions according to what might be pleasing to imagine instead of by appealing to reality.
3. **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()

   Ans:

   (a), (d), (e) → constructors
   (b), (c), (f) → selectors

4. **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.
   - **Example:**

     lst := [10, 20]

     x, y := lst
   - 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.
   - **Example:**

     lst[0]

     10

     lst[1]

     20
5. **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]

   Ans:

   (a), (e) → list

   (b), (d) → tuple

   (c), (f) → class

Part - IV (5Marks)

Answer the following questions:

1. **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 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).Notice that you don't need to know how these functions were implemented. You are assuming that someone else has defined them for us.

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

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

To enable us to implement the concrete level of our data abstraction, Some languages like Python provides a compound structure called Pair which is made up of list or Tuple. The first way to implement pairs is with the List construct.

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

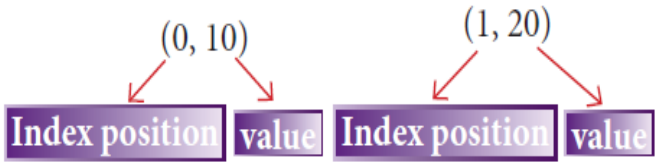
In the above example ***x*** will become10 and ***y*** will become 20.

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

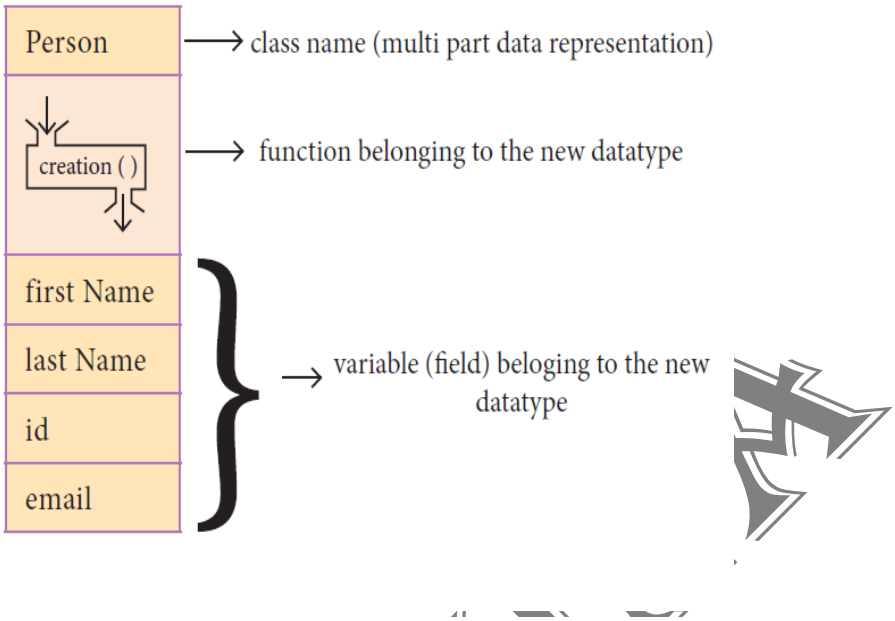
You can now represent a rational number as a pair of two integers in **pseudo code :** a numerator and a denominator.

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

3. **How will you access the multi-item. Explain with example.**
   - Lists do not allow us to do is name the various parts of a multi- item object.
   - 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, the lastName, 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:
creation( )
firstName := " "
lastName := " "
id := " "
email := " "
```



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 definition 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.
- Therefore we can define **a class as bundled data and the functions that work on that data**. From All the above example and explanation one can conclude the beauty of data abstraction is that we can treat complex data in a very simple way.

INTERIOR ONE MARK:

1. ---------- is a powerful concept in Computer Science.
   (a) data (b) function **(c) data obstruction** (d) polymorphism
2. ------ provides modularity
   (a) data **(b) data abstraction** (c) class (d) function

3. ----- is a type for objects whose behaviour is defined by a set of values and set of operations.
   (a) data (b) information **(c) ADT** (d) algorithm
4. ----- does not specify how data will be organized in memory
   (a) data **(b) ADT** (c) algorithm (d) class
5. How many types of functions are needed to facilitate data abstraction
   (a) 3 **(b) 2** (c) 4 (d) 5
6. Choose the incorrect Pair:
   a) modularity - Splitting a program
   b) Constructor - data abstraction
   c) Concrete data - definition is unknown
   d) abstract data type - only operations to be performed
7. ----- is a collection of constructors and selector.
   (a) data abstraction (b) encapsulation (c) class (d) object
8. Any program consist of ------ parts.
   (a)3 **(b) 2** (c) 4 (d) 5
9. **Assertion** : data abstraction facilitated by using constructor and selector.
   Reason : Selectors are functions that retrieve information from the data type.
   a) Assertion is correct but reason is not exact for assertion
   b) Both are wrong
   c) reason is correct but assertion is wrong
   d) Both are correct
10. ------ is the formation of beliefs and making decisions.
    (a) Wishful thinking (b) decision making (c) interface (d) implementation
11. Python provides a compound structure called ------
    (a) list **(b) pair** (c) Tuple (d) none
12. An expression in the list is called ------
    (a) pair (b) element **(c) list literal** (d) data
13. ----- can store multiple values
    (a) Tuple **(b) list** (c) element (d) pair
14. There are ------ ways to access the elements from the list
    (a) 3 **(b) 2** (c) 4 (d) 6
15. Any way of bundling two values together into one can be considered as a -------
    (a) list **(b) pair** (c) Tuple (d) data
16. ----- is a compound data type that holds two other pieces of data
    (a) list **(b) pair** (c) class (d) Tuple
17. In which, cannot change the elements, once it is assigned
    (a) Tuple (b) pair (c) list (d) element
18. In ------ we can change the lements.
    (a) Tuple **(b) list** (c) pair (d) element

19. Choose the correct statements.
a)Constructors are functions that build the abstract data type.
b) a class that not includes the data and functions.
c) list is called array
d) We can change the duple elements

20. ------- as bundled data and the functions that work on that data.
(a) object **(b) class** (c) interface (d) list

21. Match the following

(a) square bracket	-	retrieve information
(b) parenthesis	-	multiple assignment
(c) access of element	-	tuple
(d) selector	-	list

(a) 4 3 2 1 (b) 2 3 4 1
(c) 4 3 1 2 (d) 3 4 1 2

INTERIOR 2 MARKS:

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

INTERIOR 3 MARKS:

1. **What are the two parts of a program? explain**
The two parts of a program are, the part that operates on abstract data and the part that defines a concrete representation,
2. **What are the different ways to access the elements in the list explain?**
The elements of a list can be accessed in two ways.
   - ❖ multiple assignment
   - ❖ element selection operator
3. **Define class**
A class as bundled data and the functions that work on that data.

INTERIOR 5 MARKS:

1. Explain about the representation of ADT using rational numbers with example **(Page.No:13)**
2. Explain briefly the concept of constructor and selector and give example- **(Page.No:12)**

CHAPTER-3 SCOPING
Part - I (1 Mark)

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

Part - II (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.
2. **Why scope should be used for variable. State the reason.**
   Every variable defined in a program has global scope. Once defined, every part of the program can access that variable. But it is a good practice to limit a variable's scope to a single definition. 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?**
   - Programming languages keeps track of all these mappings with namespaces.
   - Namespaces are containers for mapping names of variables to objects
   - **Example:** name:=object
   -

5. **How Python represents the private and protected Access specifies?**
   - Python doesn't have any mechanism that effectively restricts access to any instance variable or method.
   - Python prescribes a convention of prefixing the name of the variable or method with single or double underscore to emulate the behavior of protected and private access specifies.

Part - III (3 Marks)

Answer the following questions:

1. **Define Local scope with an example.**
   - 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()
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.
   - This means, 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
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.**
   - All programming languages permit functions to be nested. A function (method) with in another function is called nested function.

- 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.

Example

```
1. Disp():
2. a:=10
3. Disp1():
4. print a
5. Disp1()
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.

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

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

Ans: b:=Blue →local scope
g:=Green →enclosed scope
color:=Red →global scope

Output: Blue
Green
Blue
Red

Part – IV (5Marks)

Answer the following questions:

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

Types of Variable Scope:

There are 4 types of Variable Scope, let's discuss them one by one:

- **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()
```

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.

- **Example**

```
1. a:=10
2. Disp():
3. a:=7
4. print a
5. Disp()
6. 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.
- **Enclosed Scope**
  - All programming languages permit functions to be nested. A function (method) with in another function is called nested function.
  - 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.

Example

```
1. Disp():
2. a:=10
3. Disp1():
4. print a
5. Disp1()
6. print a
7. Disp()
```

Output of the Program

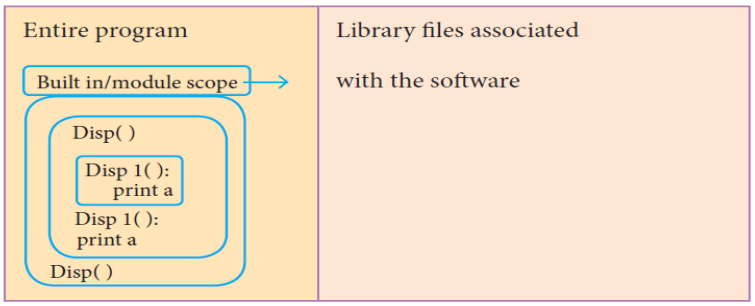
```
10
10
```

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().

- **Built-in Scope**

➢ Finally, we discuss about the widest 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.



➢ Normally only Functions or modules come along with the software, as packages. Therefore they will come under Built in scope.

(OR)

LEGB rule:

➢ Scope also defines the order in which variables have to be mapped to the object in order to obtain the value. Let us take a simple example as shown below:

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

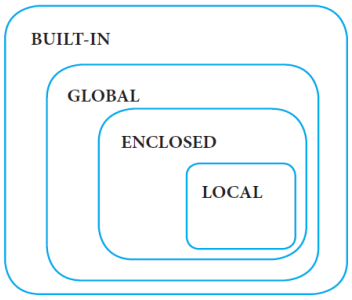
➢ When the above statements are executed the statement (4) and (5) display the result as

Output

outer x variable
inner x variable

- Above statements give different outputs because the same variable name x resides in different scopes, one inside the function display() and the other in the upper level.
- The value 'outer x variable' is printed when x is referenced outside the function definition. Whereas when display() gets executed, 'inner x variable' is printed which is the x value inside the function definition.
- From the above example, we can guess that there is a rule followed, in order to decide from which scope a variable has to be picked.
- The **LEGB** rule is used to decide the order in which the scopes are to be searched for scope resolution. The scopes are listed below in terms of hierarchy (highest to lowest).

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



2. Write any Five Characteristics of Modules.

The following are the desirable characteristics of a module.

- 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.

- 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.

INTERIOR ONE MARK:

1. ------ refers to the visibility of variables, parameters and functions in one part of program to another part of the same program.
   (a) Visible **(b) Scope** (c) mode (d) variable
2. Every variable defined in a program has ------ scope.
   (a) local **(b) global** (c) variable (d) class
3. ----- are called references, addresses, pointers.
   (a) variable (b) scope (c) visible (d) life time
4. The process of binding a variable name with an object is called -----
   (a) variable **(b) mapping** (c) assign (d) value
5. ----- is used to programming languages to map the variable and object.
   (a) : (b) = **(c) : =** (d) –
6. ----- are containers for mapping names of variables to object.
   (a) value (b) variable (c) memory **(d) namespaces**
7. The ------ of a variable is that part of the code where it is visible.
   (a) visible **(b) scope** (c) life time (d) access
8. The duration for which a variable is alive is called ------
   (a) visible **(b) life time** (c) scope (d) variable
9. The ----- is used to decide the order in which the scopes are to be searched for scope resolution.
   (a) LGEB **(b) LEGB** (c) LG (d) GE
10. ------ defined inside function class
    (a) global **(b) local** (c) built in (d) enclosed
11. ------ is the nested function concept.
    (a) local **(b) enclosed** (c) global (d) none
12. ------ is defined at the uppermost level.
    (a) local **(b) global** (c) bail in (d) enclosed
13. There are ---- type of variable scope.
    (a) 3 **(b) 4** (c) 5 (d) 7
14. A function with another function is called ------ function.
    (a) nested (b) upper (c) local (d) global
15. Built in scope is also called as ------
    (a) local scope (b) global scope **(c) module scope** (d) none

16. In which scope has all the names that are preloaded into the program scope when are start the compiler (or) interpreter.
(a) local **(b) built-in** (c) global (d) enclosed

17. Choose the correct statement:
a) Namespaces are contains for mapping names of variables to object.
b) Scope refers to functions.
c) LEGB role is used to decide the orders in scope of variables.
d) There are 5 types of scope.
a) a, c is correct b) a, b, c is correct c) a, d is correct d) only a is correct

18. ----- is a part of program.
(a) data (b) functions (c) object **(d) module**

19. ------ does not have any mechanism that effectively restricts access to any instance variable or method.
(a) C **(b) Python** (c) C++ (d) java

20. All members in a python class are ----- by default.
(a) public (b) private (c) protected (d) none

21. All members in C++, java are ------ by default.
(a) public **(b) private** (c) protected (d) none

22. Choose the incorrect pair:
a) LEGB - lowest to highest
b) modular programming - debug independently
c) default access specifies - Public
d) local scope - current function

23. Any member can be accessed from outside the class environment in ------
(a) C (b) java **(c) Python** (d) C++

24. In python, a variable prefixing with single underscore denotes ------
(a) private (b) public **(c) protected** (d) class

25. A variable prefixing with double underscore denotes -------
(a) public **(b) private** (c) protected (d) class

26. There are ----- types of access specifies.
(a) 4 **(b) 3** (c) 5 (d) 7

27. Match the following
(a) local - Preloaded in library function
(b) global - nested function
(c) enclosed - within current function
(d) built-in - Outside the function
(a) 3 4 2 1 (b) 4 3 1 2
(c) 3 4 1 2 (d) 2 3 4 1

INTERIOR 2 MARKS:

1. **What is modular programming**

   A program can be divided into small functional modules that work together to get the output. The process of subdividing a computer program into separate sub-programs is called Modular programming.

2. **Name the access specifies used in programming language.**
   - ❖ Private
   - ❖ Protected
   - ❖ Public

INTERIOR 3 MARKS:

1. **Define data encapsulation.**

   This arrangement of private instance variables and public methods ensures the principle of data encapsulation.

2. **Write about life time of a variable.**

   The duration for which a variable is alive is called its 'life time'.

INTERIOR 5 MARKS:

1. Explain about access control in briefly. **(Page.No:26)**

CHAPTER- 4 ALGORITHMIC STRATEGIES

Part - I (1 Mark)

Choose the best answer:

1. The word comes from the name of a Persian mathematician Abu Jafar Mohammed ibn 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 (nlogn) (C) θ (n2) (D) θ (n(logn) 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) Memoization (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) Memoization (D) Mapping

Part - II (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. An algorithm can be implemented in any suitable programming language.
2. **Define Pseudo code.**
   - Pseudo code is an informal high level description of the operations principle of a computer program or other algorithm.
   - It uses the structural conventions of a normal programming language, but is indented for human reading rather than machine reading.
3. **Who is an Algorist?**
   - Algorism is the technique of performing basic arithmetic by writing numbers in place value form and applying a set of memorized rules and facts to the digits.
   - One who practices algorism is known as an algorist.
4. **What is Sorting?**
   - Sorting is any process of arranging information or data in an ordered sequence either in ascending or in descending order.
5. **What is searching? Write its types.**
   - Searching is designed to check for an element or retrieve an element from any data structure where it is stored.

   Types:
   - Linear (or) sequential search
   - Binary (or) half interval search

Part - III (3 Marks)

Answer the following **questions:**

1. **List the characteristics of an algorithm.**
   - Input
   - Output
   - Finiteness
   - Definiteness
   - Effectiveness

- Correctness
- Simplicity
- Unambiguous
- Feasibility
- Portable
- Independent

➢ Input- Zero or more quantities to be supplied
➢ Output – At least one quantity is produced
➢ Simplicity – Easy to implement

2. Discuss about Algorithmic complexity and its types.

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.

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

➢ **fixed part**
➢ **Variable part**

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

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.

4. Write a note on asymptotic notation.

➢ Asymptotic Notations are languages that uses 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 Bi O is used to describe the upper bound (worst - case) of a 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 is an algorithmic design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.
➢ Dynamic programming approach is similar to divide and conquer.
➢ The given problem will be divided into smaller overlapping 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.

Part - IV (5Marks)

Answer the following questions:

1. **Explain the characteristics of an algorithm.**

Input	Zero or more quantities to be supplied
Output	At least one quantity is produce.
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 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.

Pseudo code

(i) Traverse the array using for loop

(ii)In every iteration, compare the target search key value with the current value of the list.

- If the values match, display the current index and value of the array
- If the values do not match, move on to the next array element.

(iii) 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 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 Step1.

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.

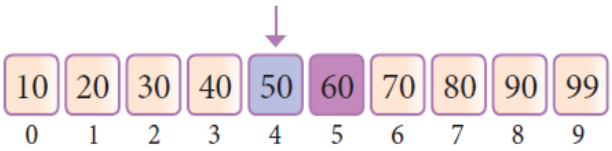
Binary Search Working principles

- 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.



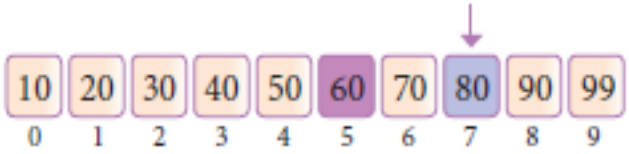
Now compare the search element with the value stored at mid value location 4. The value stored at location or index 4 is 50, which is not match with search element. As the search value 60 is greater than 50.

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

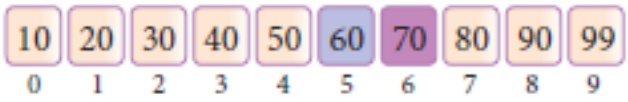
low to mid + 1

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

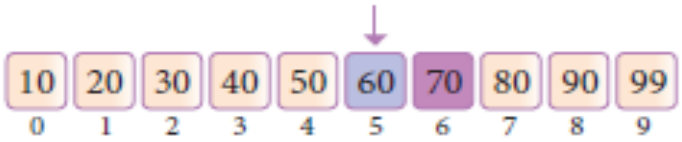


The search element still not found. Hence, we calculated the mid again by using the formula.

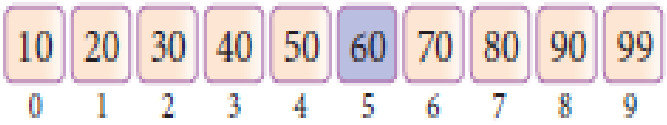
high = mid -1

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 :

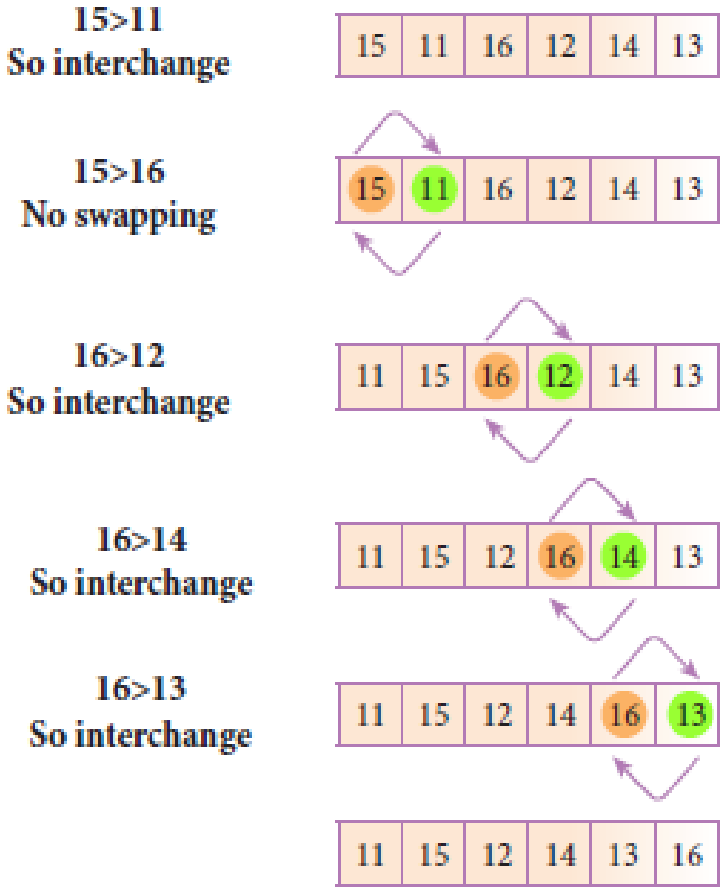
- Bubble sort is a simple sorting algorithm. The algorithm starts at the beginning of the list of values stored in an array.
- It compares each pair of adjacent elements and swaps them if they are in the unsorted order. This comparison and passed to be continued until no swaps are needed, which indicates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

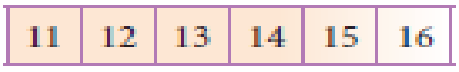
Pseudo code

- 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.

Example : 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 is an algorithmic design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions. 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 Memoization.

- **Fibonacci Series – An example**

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

Fibonacci series satisfies the following conditions:

Fib n = Fibn-1 + Fibn-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**
- **Example: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

- **Example:** generate fibonacci series upto 10 digits

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

INTERIOR ONE MARKS:

1. An ------ is a finite set of instructions to accomplish a particular task.
   (a) Program **(b) algorithm** (c) instruction (d) data
2. ------ is a step-by-step procedure for solving a given problem.
   (a) program **(b) algorithm** (c) instruction (d) data
3. ----- can be implemented in any suitable programming language.
   (a) algorithm (b) program (c) instruction (d) data
4. Data are maintained and manipulated effectively through ------
   (a) algorithm **(b) data structures** (c) instruction (d) program
5. ----- can be developed to store, manipulate and retrieve data from data structure.
   (a) algorithm (b) data structure (c) instruction (d) program
6. Pick out odd one from the following
   (a) array (b) structure (c) Tuples **(d) information**
7. ----- to search an item in a data structure.
   (a) sort **(b) search** (c) insert (d) delete
8. ----- to sort items in a certain order using the methods such as bubble sort.
   (a) sort (b) search (c) insert (d) Delete

9. ----- is used to insert an item in a data structure.
(a) sort (b) delete **(c) insert** (d) search

10. ------- is to delete an existing item is in a data structure
(a) sort **(b) delete** (c) insert (d) search

11. ----- to update an existing item is in a data structure.
(a) sort **(b) update** (c) insert (d) search

12. The way of defining an algorithm is called -------
(a) instruction **(b) algorithmic strategy** (c) solution (d) program

13. The word algorithm comes from the name of a ----- author.
(a) book (b) German **(c) Persian** (d) mathematics

14. The word ---- has come to refer to a method to solve a problem.
(a) algorithm (b) program (c) data (d) instruction

15. ----- are generic and not limited to computer alone.
(a) algorithm (b) program (c) data (d) instruction

16. An algorithm that yields expected output for a valid input is called an ------
(a) algorithm **(b) algorithmic solution**
(c) algorithmic strategy (c) program

17. The ----- of an algorithm is defined by the utilization of time and space complexity
(a) effective **(b) efficiency** (c) analysis (d) complexity

18. Analysis of an algorithm usually deals with the running and execution time of various operations involved.
(a) algorithm **(b) analysis** (c) instruction (d) program

19. The ------- time of an operation is calculated as how many programming instructions executed per operation
(a) execution **(b) running** (c) program (d) instruction

20. ------ is a theoretical performance analysis of an algorithm.
(a) priori estimates (b) posteriori testing (c) efficiency (d) algorithm

21. Efficiency of an algorithm is measured by assuming the factors.
(a) internal **(b) external** (c) priori (d) posteriori

22. ----- is called performance measurement
(a) priori estimates **(b) posteriori testing** (c) internal factor (d) time factor

23. An estimation of the time and space complexities of an algorithm for varying input sizes is called -------
(a) analysis **(b) algorithm analysis** (c) procedure (d) instruction

24. Time is measured by counting the number of key operations like comparisons in the sorting algorithm is called ------
(a) space factor **(b) Time factor** (c) Time complexity (d)space complexity

25. Space is measured by the maximum memory space required by the algorithm is called ------
(a) space factor (b) Time factor (c) Time complexity (d) efficiency

26. The ------ of an algorithm is given by the number of steps taken by the algorithm to complete the process.
(a) Time complexity (b) space complexity (c) Time factor (d) space factor

27. ------ of an algorithm is the amount of memory required to run its completion.
(a) Time complexity **(b) space complexity** (c) Time factor (d) space factor

28. ------ is defined as the total space required to store certain data and variables for an algorithm.
(a) space factor (b) Time factor **(c) fixed part** (d) variable part

29. ---- is defined as the total space required by variables
(a) space factor (b) Time factor (c) fixed part **(d)variable part**

30. For maximum efficiency of algorithm we wish to minimize ------ usage.
(a) time **(b) resource** (c) space (d) program

31. The ----- algorithm to solve a given problem is one that requires less space in memory and take less time to execute its instructions to generate output.
(a) best algorithm (b) worst algorithm (c) instruction (d) computer

32. ----- are languages that use meaningful statements about time and space complexity
(a) algorithm (b) instruction (c) procedure (d) asymptotic notation

33. Match the following :

(a) Worst case	-	(1) Big H
(b) best case	-	(2) Big O
(c) average case	-	(3) designing algorithm
(d) algorithmic strategy	-	(4) Big Ω

(a) 2 4 1 3 (b) 4 1 2 3 (c) 2 1 4 3 (d) 4 2 1 3

34. The ------- algorithm starts at the beginning of the list of values stored in an array.
(a) selections sort **(b) Bubble sort** (c) insertion sort (d) merge sort

35. ------ compares each pair of adjacent elements and swaps the if they are in the unsorted order.
(a) selection sort **(b) Bubble sort** (c) insertion sort (d) merge short

36. ---- is named for the way smaller elements “bubble” to the top of the list.
(a) selection sort **(b) Bubble sort** (c) insertion sort (d) merge short

37. ---- is simple, it is too slow and less efficient when compared to insertion sort and other sorting methods.
(a) selection sort **(b) Bubble sort** (c) insertion sort (d) merge short

38. Choose the correct pair:

a) space complexity	-	**fixed part**
b) pair	-	single structure
c) list	-	multi item object
d) Binary search	-	unsorted array

39. ----- sort improves on the performance of bubble sort by making only one exchange for every pass through the list.
(a) selection sort (b) bubble sort (c) insertion sort (d) merge sort

40.----- selects the next-smallest element and swaps into the right place for every pass.
(a) selection sort (b) bubble sort (c) insertion sort (d) merge sort

41.----- taking elements from the list one by one and inserting then in their correct position into a new sorted list.
(a) selection sort (b) bubble sort **(c) insertion sort** (d) merge sort

42.------- is an algorithmic design method.
(a) dynamic programming (b) optimization (c) algorithm (d) procedure

43.Dynamic programming approach is similar to -------
(a) selection sort (b) bubble sort **(c) divide and conquer** (d) linear search

44.------ algorithm will try to check the results of the previously solved sub-problems.
(a) selection sort **(b) dynamic** (c) divide and conquer (d) linear search

45.Dynamic algorithm uses --------
(a) divide and conquer (b) linear search (c) pseudo code **(d) memorization**

46.------ is an optimization technique used primarily to speed up computer programs.
(a) memorization (b) divide and conquer (c) pseudo code (d) Binary search

47.Pick out odd one from the following
(a) sort (b) insert (c) update **(d) function**

48.Choose the correct statements.
(a) Dynamic algorithms uses memorization
(b) Binary search is also called half-interval search algorithm
(c) linear search also called sequential search
(d) divide and conquer algorithm is used in the linear search.
(a) a only correct **(b) a, b, c is correct** (c) d is correct (d) all are correct

49.Choose the incorrect pair.
a) input - Zero or more quantities
b) correctness - error free
c) simplicity - easy to implement
d) finiteness - no limitation for number of steps

INTERIOR 2 MARKS:

1. **Define algorithmic strategy**
   The way of defining an algorithm is called algorithmic strategy.
2. **Define algorithmic solution**
   An algorithm that yields expected output for a valid input is called an algorithmic solution.
3. **Define priori estimates**
   This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.
4. **Define posteriori testing**
   This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.
5. **What are the two different phases in analysis of algorithm**
   (i)priori estimates (ii) posteriori testing

INTERIOR 3 MARKS:

1. **Define algorithm analysis.**
   An estimation of the time and space complexities of an algorithm for varying input sizes is called algorithm analysis.
2. **What are the two components of space complexity? Explain.**

❖ **A fixed part:** It 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:** It 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.

3. **What is meant by best algorithm explain.**
   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.
4. **What are the types of asymptotic notations? Explain.**
   (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 Bi O is used to describe the upper bound (worst - case) of a 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. **Define memorization.**
   Memorization or memoisation is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again.

INTERIOR 5 MARKS:

1. Explain the difference between algorithm and program.-**(Page.No:33)**
2. Explain about the complexity of an algorithm.**-(Page.No:34)**
3. Explain about analysis of algorithm.-**(Page.No:33)**