[HALF YEARLY EXAMINATION KEY ANSWER -2024-2025]

STD: XII – COMPUTER SCIENCE – DR SURESH MATRIC HSS – RAMANATHAPURAM (DIST)

I (One word)

1.b 2.d 3.a 4.c 5.a 6.b 7.c 8.c 9.b 10.d 11.a 12.b 13.a 14.b 15.b

II (Two Marks) (Q.no 24 is compulsory)

16

- ❖ 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.

17

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

18 Namespaces are containers for mapping names of variables to objects. **<u>Example:</u>** a:=5

19

- ❖ 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.

20 Interactive mode and Script mode are the two modes **that can be used to test Python Program.**

21 1.Sequential **2.**Alternative or Branching 3.Iterative or Looping

22 It avoids repetition and makes high degree of code reusing. It provides better modularity for your application

23

- ❖ A Set is a mutable and an unordered collection of elements without duplicates.
- ❖ That means the elements within a set cannot be repeated.

24 **Output:** I love computer science (Write 4 times)

III (Three Marks) (Q.no 33 is compulsory)

25

- ❖ Variables defined inside a class are called as "Class Variable" and functions are called as "Methods".
- ❖ Class variable and methods are together known as members of the class.

Syntax: **<u>Example:</u>**

```
class class_name:          class Sample:
    statement_1                x = 10 → class variable
    statement_2               def disp(self): → method
    …………..                     print(Sample.x)
     …………..                    s = Sample()
    statement_n                s.disp()
```

26

- ❖ Cross product is a way of combining two relations.
- ❖ The resulting relation contains, both relations being combined.
- ❖ This type of operation is helpful to merge columns from two relations.

<u>Example</u>: A x B means A times B, where the relation A and B have different attributes.

<u>Table A</u>

Roll No	Name
11C01	Santhosh
11C02	Sujith

<u>Table B</u>

Sub Code	Subject
123	CS
456	CA

Cartesian product : <u>Table A x Table B</u>

RollNo	Name	Sub Code	Subject
11C01	Santhosh	123	CS
11C01	Santhosh	456	CA
11C02	Sujith	123	CS
11C02	Sujith	456	CA

27 1.Create Command: To create tables in the database. 2.Alter Command: Alters the structure of the database.
3.Drop Command: Delete tables from database
4.Truncate: Remove all records from a table, also release the space occupied by those records.

28

PYTHON	C++
❖ Python is typically an "interpreted" language	❖ C++ is typically a "compiled" language
❖ Python is a dynamic-typed language	❖ ++ is compiled statically typed language
❖ Data type is not required while declaring variable	❖ Data type is required while declaring variable
❖ It can act both as scripting and general purpose language	❖ It is a general purpose language.

29

- ❖ Asymptotic Notations are languages that uses meaningful statements about time and space complexity.

(i) Big O → worst-case of algorithm (ii) Big Ω → best-case of algorithm
(iii) Big Θ → average-case (complexity of an algorithm Lower bound = upper bound

30

- ❖ '=' is a simple assignment operator to assign values to variable.
- ❖ There are various compound operators in Python like **+=, -=, *=, /=, %=,**= and** //= are also available.

<u>Example:</u>

a=5 # assigns the value 5 to a a,b=5,10 # assigns the value 5 to a and 10 to b
a+=2 # a=a+2, add 2 to the value of „a" and stores the result in 'a'(Left hand operator)

31

1.	**capitalize( )**	❖ Used to capitalize the first character of the string	>>> city="chennai" print(city.capitalize()) **Output**: Chennai
2.	**Swap case( )**	❖ It will change case of every character to its opposite case vice-versa.	>>> str1="tAmiLNaDu" print(str1.swapcase()) **Output**: *TaMIlnAdU*

32

- ❖ **Union (|)** : It includes all elements from two or more sets.
- ❖ **Intersection (&)** : It includes the common elements in two sets.
- ❖ **Difference (-)** : It includes all elements that are in first set (say set A) but not in the second set (say set B).
- ❖ **Symmetric difference** (^): It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.

33 **Output:** Aate Bate Cate Date Eate Fate Gate Hate

IV **(Five Marks)**

38a a) List b) Tuple c) Class d) Tuple e) List

Or **<u>Binary search:</u>**

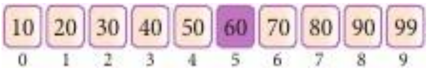
- ❖ Binary search also called half-interval search algorithm. It finds the position of a search element within a sorted array.
- ❖ The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.

<u>Pseudo code for Binary search:</u>

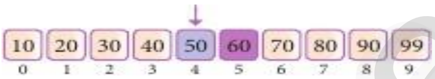
```
1.  Start with the middle element:
a)  If the search element is equal to the middle element of the array, then return the index of the middle element.
b)  If not, then compare the middle element with the search value,
c)  If (Search element > number in the middle index), then select the elements to the right side of the middle index, and go to Step-1.
d)  If (Search element < 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.
```

<u>Binary Search Working principles with example:</u>

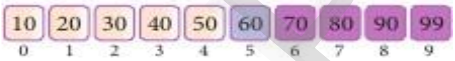
- ❖ List of elements in an array must be sorted first for Binary search.
- ❖ The array is being sorted in the given example and it is suitable to do the binary search algorithm.
- ❖ Let us assume that the **search element is 60** and we need to search the location or index of search element 60 using binary search.



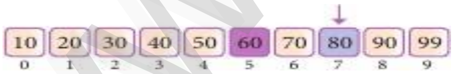
- ❖ First, we find index of middle element of the array by using this formula:
  mid = low + (high - low) / 2
- ❖ Here it is, 0 + (9 - 0) / 2 = 4. So, 4 is the mid value of the array.



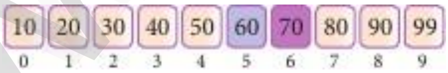
- ❖ Now compare the search element with the value stored at mid value location 4.
- ❖ The value stored at location or index 4 is 50, which is not match with search element.
- ❖ As the search value 60 is greater than 50.



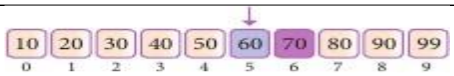
- ❖ Now we change our low to mid + 1 and find the new mid value again using the formula.
  low = mid + 1
  mid = low + (high - low) / 2
- ❖ Our new mid is 7 now. We compare the value stored at location 7 with our target value 60.



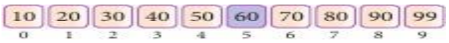
- ❖ The value stored at location or index 7 is not a match with search element, rather it is more than what we are looking for.
- ❖ So, the search element must be in the lower part from the current mid value location



- ❖ 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.

34 a

- ❖ **Parameters** are the variables in a function definition
- ❖ **Arguments** are the values which are passed to a function definition.
- ❖ Two types of parameter passing are,

    i) Parameter Without Type ii) Parameter With Type

<u>(i) Parameter Without Type:</u>

<u>Let's see an example of a function definition of Parameter Without Type:</u>

```
(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** passed **to** the variable **'b'** is the **argument**.
- ❖ The precondition **(requires)** and post condition **(returns)** of the function is given.
- ❖ We have not mentioned any types: **(data types)**.
- ❖ This is called **parameter without type.**
- ❖ In the above function definition the expression has type **'int',** so the function's return type alsobe '**int'** by implicit.

<u>(ii) Parameter With Type:</u>

<u>Now let us write the same function definition with types,</u>

```
(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)
```

- ❖ In this example we have explicitly annotating the types of argument and return type as**'int'**.
- ❖ Here, when we write thetype annotations for 'a' and 'b' the parentheses are mandatory.
- ❖ This is the way passing parameter with type which helps the compiler to easily infer them.

OR

<u>i) Python's sys module</u>:

- ❖ This module provides access to some variables used by the interpreter and to functions that interact strongly with the interpreter.

<u>sys. argv:</u>

- ❖ sys.argv is the list of command-line arguments passed to the Python program.
- ❖ **argv contains** all the items that come via the command-line input, it's basically a list holding the command-line arguments of the program.

<u>ii) Python's OS Module</u> :

- ❖ The OS module in Python provides a way of using operating system dependent functionality.
- ❖ The functions that the OS module allows you to interface with the Windows operating system where Python is running on.

<u>os.system():</u>

- ❖ Execute the C++ compiling command (a string contains Unix, C command which also supports C++ command) in the shell (Here it is Command Window).

<u>iii) Python getopt module :</u>

- ❖ The getopt module of Python helps you to parse (split) command-line options and arguments.
- ❖ This module provides two functions to enable command-line argument parsing.
- ❖ This method parses command-line options and parameter list.

<u>Syntax :</u> <opts>,<args>=getopt.getopt(argv, options, [long_options])

35 a

<u>I. List:</u>

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

<u>1. Multiple Assignment:</u>

- ❖ Which unpacks a list into its elements and binds each element to a different name.

Example: lst := [10, 20] x, y := lst **x** will become10 and **y** will become 20.

2. Element Selection Operator:

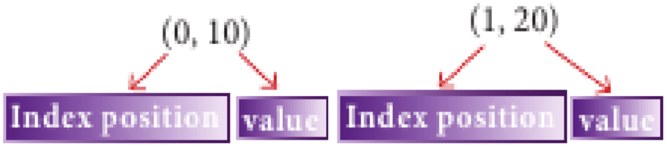
- ❖ It is expressed using square brackets.
- ❖ Unlike a list literal, a square-brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.

Example: lst[0] 10 lst[1] 20

II. Pair:

- ❖ 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[(0,10),(1,20)]



OR

Excel	CSV
❖ Excel is a binary file that holds information about all the worksheets in a file, including both content and formatting.	❖ CSV format is a plain text format with a series of values separated by commas.
❖ XLS files can only be read by applications that have been especially written to read their format, and can only be written in the same way.	❖ CSV can be opened with any text editor in Windows like notepad, MS Excel, Open Office, etc.
❖ Excel is a spreadsheet that saves files into its own proprietary format viz. xls or xlsx	❖ CSV is a format for saving tabular information into a delimited text file with extension .csv
❖ Excel consumes more memory while importing data	❖ Importing CSV files can be much faster, and it also consumes less memory.

36 a

Tuple:

- ❖ Tuples consists of a number of values separated by comma and enclosed within parentheses.
- ❖ Tuple is similar to list, values in a list can be changed but not in a tuple.

Nested Tuples:

- ❖ In Python, a tuple can be defined inside another tuple; called Nested tuple.
- ❖ In a nested tuple, each tuple is considered as an element.
- ❖ The for loop will be useful to access all the elements in a nested tuple.

Example:

```
Toppers = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H", 97.5), ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8))
for i in Toppers:
print(i)
```

Output:

```
('Vinodini', 'XII-F', 98.7)
('Soundarya', 'XII-H', 97.5)
('Tharani', 'XII-F', 95.3)
('Saisri', 'XII-G', 93.8)
```

OR

Basis of Comparison	DBMS	RDBMS
Expansion	Database Management System	Relational Data Base Management System
Data storage	Navigational model ie data by linked records	Relational model.ie data in tables as row and column
Data redundancy	Exhibit	Not Present
Normalization	Not performed	RDBMS uses normalization to reduce redundancy.
Data access	Consumes more time	Faster, compared to DBMS
Keys and indexes	Does not use.	Used to establish relationship. Keys are used in RDBMS.
Transaction management	Inefficient, Error prone and insecure.	Efficient and secure.
Distributed Databases	Not supported	Supported by RDBMS
Example	Dbase, FoxPro.	SQL server, Oracle, my sql, Maria DB, SQLite.

37 a

List and range ( ) function:

- ❖ The range ( ) is a function used to generate a series of values in Python.
- ❖ Using range ( ) function, you can create list with series of values.
- ❖ The range ( ) function has three arguments.

Syntax of range ( ) function:

range (start value, end value, step value) where,

- ❖ **Start value** – beginning value of series. Zero is the default beginning value.
- ❖ **End value** – upper limit of series. Python takes the ending value as upper limit – 1.
- ❖ **Step value** – It is an optional argument, which is used to generate different interval of values.

Example : Generating whole numbers up to 10 **Output**

```
for x in range (1, 11):                              1 to 10
print(x)
```

Creating a list with series of values :

- ❖ Using the range () function, you can create a list with series of values.
- ❖ To convert the result of range ( ) function into list, we need one more function called list ().
- ❖ The list () function makes the result of range ( ) as a list.

Syntax: List_Varibale = list (range ( ))

Example: **Output**

```
Even_List = list(range(2,11,2))                 [2, 4, 6, 8, 10]
>>> print(Even_List)
```

OR

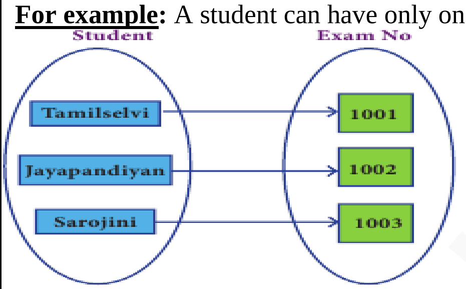
Types of relationships used in a database:

1. One-to-One Relationship 2. One-to-Many Relationship 3. Many-to-One Relationship 4. Many-to-Many

1. One-to-One Relationship :

- ❖ In One-to-One Relationship, one entity is related with only one other entity.
- ❖ One row in a table is linked with only one row in another table and vice versa.

For example: A student can have only one exam number

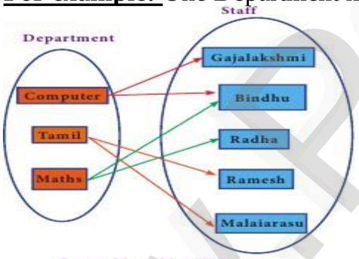


One to one Relationships

2. One-to-Many Relationship:

- ❖ In One-to-Many relationship, one entity is related to many other entities.
- ❖ One row in a table A is linked to many rows in a table B, but one row in a table B is linked to only one row in table A.

For example: One Department has many staff members.



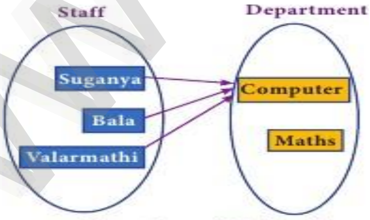
One to Many Mapping

3. Many-to-One Relationship:

- ❖ In Many-to-One Relationship, many entities can be related with only one in the other entity.

For example: A number of staff members working in one Department.

- ❖ Multiple rows in staff members table is related with only one row in Department table.



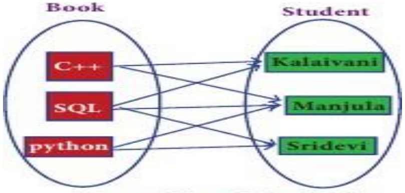
Many to one Relationship

4. Many-to-Many Relationship:

❖ A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table.

Example 3: Books and Student.

❖ Many Books in a Library are issued to many students.



Many to Many Relationship