Class : XII EM **FIRST MID TERM EXAM 2019 ANSWER KEY** **DATE : 05.08.2019**

I)	**1)**	Function	**2)**	List
	3)	Public	**4)**	O(n)
	5)	1991	**6)**	Whitespace
	7)	3	**8)**	Jump
	9)	Lambda	**10)**	a

II)	**11)**	**Characteristics of interface.** • The class temples specify the interfaces to enables an object to be created and operated properly. • An object's attributes and behaviour is controlled by sending functions to the object.
	12)	**Different between tuples and list** • The difference between the two is that you cannot change the elements of a tuple once it is assigned. • In a list, elements can be changed.
	13)	**Scope :** • Scope refers to the visibility of variables, Parameters and functions in one parts of a program. In other words, which parts of your program can see or use it.
	14)	**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 algorithm uses Memoization.

	15)*	**Ternary operator**
		• Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false.
		• It simply allows testing a condition in a single line replacing the multiline if else making the code compact.
		The syntax conditional operator is,
		Variable name =[on_true]if[Test expression]else[on_false]
		Example:
		min= 50 if 49<50 else 70 # min = 50
		min= 50 if 49>50 else 70 # min =70

	16)	Computer science End of the Program
	17)	**Advantages of functions** • It avoids repetition and makes high degree of code reusing. • It provides better modularity for your application.
III)	**18)**	**Pure functions** • Pure functions are functions which will give exact result when the same arguments are passed • The return value of the pure functions solely depends on its arguments passed. • 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. • They do not modify the arguments which are passed to them. **Example:** let square x return: x * x
	19)	**Strategy for designing programs** • We are using 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 imaging instead of by appealing to reality.

20) Modular programming

- A module is a part of a program. Programs are composed of one or more independently developed modules.
- The process of subdividing a computer program into separate sub-programs is called Modular programming.

Usage :

1. Less code to be written.
2. A single procedure can be developed for reuse, eliminating the need to retype the code many times.
3. Programs can be designed more easily because a small team deals with only a small part of the entire code.
4. Modular programming allows many programmers to collaborate on the same application.
5. The code is stored across multiple files.

21) Asymptotic notations

- Asymptotic notations are languages that uses meaningful statement about time and space complexity. The following three asymptic notations are mostly used to represent time complexity of algorithm.

Big O **Big Ω** **Big Θ**

22)*

```
defoddeven(a):
    if (a0==2%):
        return 1
    else:
        return 0
num = int(input("Enter a number: "))
if (oddeven(num)==1):
    print("The given number is Even")
elif (oddeven(num)==0):
    print("The given number is Odd")
```

23) Literals

Literal is a law data given in a variable or constant. In Python, there are various types of literals

- Numeric
- String
- Boolean

Numeric Literals

- Numeric Literals consists of digits and are immutable (Unchangeable). Numeric literals can belong 3 different numerical types integer, float and complex.

String Literals

- In python a String literal is a sequence of character surrounded by quotes. Python supports single, double and triple quotes for a string.

Boolean Literals

- A Boolean literal can have any the two values: True or False.

24) Mathematic functions

floor ( )	Returns the largest integer less than or equal to x	math.floor (x)
ceil ( )	Returns the smallest integer greater than or equal to x	math.ceil (x)
sqrt ( )	Returns the square root of x Note: x must be greater than 0 (zero)	sqrt (x)

IV) 25) A) What are called parameters and write a note on

Parameters

Parameters are the variables in a function definition

Parameter without type

```
Let us see an example of a function definition:
(requires: b>=0)
(return: 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. In the above function definition,

- if expression can return 1 in the then b 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

```
(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.
- This is useful on times when you get a type error from the compiler that doesn't make sense.
- Explicitly annotating the type can help with debugging such an error message.

25) B) **LEGB rule with example.**

Local scope

- 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.

Examples	**Output**
1. Disp() :	7
2. a :=7	
3. print a	
4. Disp ( )	

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

Global Scope

- A variable which is declared outside of all the functions in a program is known as global variables. Global variable can be accessed inside or outside of all the functions in a program.

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

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

- A functions (method) with in another functions is called nested function.
- A variable which is declared inside a function which contains another functions definition with in it, the inner functions can also access the variable of the outer functions.
- 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.

Consider the following example

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

In 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

```
Entire program          Output
Built in/module scope   10
Disp( )                 10
a:=10
Disp1():
Print a
Disp1():
print a
Disp( )
```

The built-in scope has all the means 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 languages 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.

26) A) Selection Sort with example.

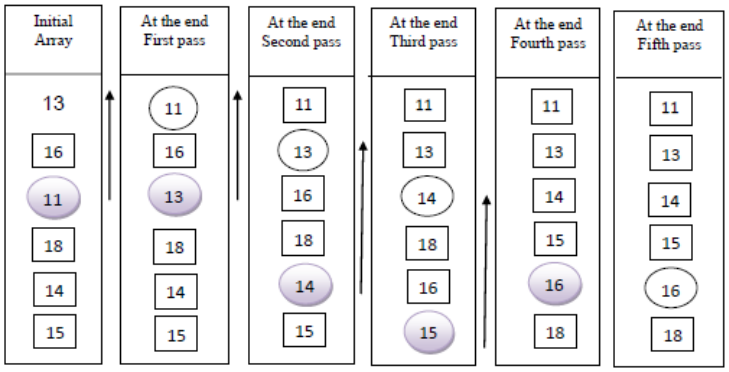
- The selection sort is a simple sorting algorithm. This algorithm will first find the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue 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

1. 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.
2. 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
3.Now replace the second smallest identified in step 2 at the second position in the original array, or also called first position in the sub array.
4. This repeated, until the array is completely stored.

Let's consider an array with values {13, 16, 11, 18, 14, 15}

Below. We have a pictorial representation of how selection sort will sort the given array.



1. In the first pass, the smallest element will be 11, so it will be placed at the first position.
2. 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.
3. Then leaving 11 and 13, we will search for the next smallest element from the rest of the elements and put it at third position.
4. Keep doing this until array is sorted.

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

26) B) Input () and Print ()function with example.

- A program needs to interact with the user to accomplish the desired task; this can be achieved using Input-Output functions.
- The input() function helps to enter data at run time by the user and the output function print() is used to display the result of the program on the screen after execution.

The print () function

- In python, the print () function is used to display result on the screen. The syntax for print () is as following:

The syntax for print ( ) as follow:
print ("string to be displayed as output")
print (variable)
print ("string to be displayed as output",variable)
print ("string1", variable,"string2", variable, "string3".....)

Example

```
>>>print ("welcome to python programming")
Welcome to python programming
>>> x=5
>>> y=6
>>> z=x+y
>>>print(z)
11
```

- ➢ The print ( ) evaluates the expression before printing it on the monitor.
- ➢ The print ( ) displays an entire statement which is specified within print ( ).
- ➢ Comma (,) is used as separator in print ( ) to print more than one item.

Input () function

- In python, input ( ) function is used to accept data as input at run time. The syntax for input() function is,

Variable = input ("prompt string")

- ➢ Prompt string in the syntax is a statement or message to the user, to know what input can be given.
- ➢ If a prompt string is used, it is displayed on the monitor; the user can provide expected data from the input device.
- ➢ input () takes whatever is typed from the keyboard and stores the entered data in the given variable. If prompt string is not given in input( ) no message is displayed on the screen, the user will not know what is to be typed as input.

- The input ( ) accepts all data as string or characters but not as numbers
- If a numerical value is entered, the input values should be explicitly converted into numeric data type.
- The int( ) function is used to convert string data as integer data explicitly.

Example :

```
x = int (input("Enter Number 1:"))
y = int(input("Enter Number 2:"))
print ("The sum=", x+y)
```

Output

```
Enter Number 1: 34
Enter Number 2: 54
The sum = 90
```

Looping Statement

- Iteration or loop are used in situation when the user need to execute a block of code several of times or till the condition is satisfied. A loop statement allows to execute a statement or group of statements multiple times.

for loop is the most comfortable loop. It is also an entry check loop. The condition is checked in the beginning and the body of the loop(statements-block 1) is executed if it is only True otherwise the loop is not executed.

Syntax:

```
for counter_variable in sequence:
    statements-block 1
[else: # optional block
    statements-block 2]
```

- The counter_variable mentioned in the syntax is similar to the control variable that we
- used in the for loop of C++ and the sequence refers to the initial, final and increment value.
- Usually in Python, for loop uses the range() function in the sequence to specify the initial, final and increment values. range() generates a list of values starting from start till stop-1.

The syntax of range() is as follows:

```
range (start,stop,[step])
```

Where,
start – refers to the initial value
stop – refers to the final value
step – refers to increment value, this is optional part.

Example :

```
for i in range (2,10,2):
    print (i, end=' ')
```

Output:

```
2 4 6 8
```

27) B)

Types of Functions:

- User-defined Functions
- Built-in Functions
- Lambda Functions
- ecursion Functions

Functions	Description
User-defined functions	Functions defined by the users themselves.
Built-in functions	Functions that are inbuilt with in Python.
Lambda functions	Functions that are anonymous un-named function.
Recursion functions	Functions that calls itself is known as recursive.