

Sample Question Paper-B

1. (a) Software is a collection of programs and related documentation. (1)
(b) ASCII supports 128 characters. (1)
(c) Dynamic Random Access Memory (1)
(d) def multiplication Table () :
 for i in range (5, 10) :
 for j in range (1, 11) :
 mul = i*j
 Print mul,
 Print.
(2)
2. (a) (i) Physical Level
(ii) Logical Level
(iii) View Level (1)
(b) Network DataBase Management System is a type of file management system in which the files are arranged in form of a network with linkages present between them . (1)
(c) (i) It is open source
(ii) It has a powerful security feature. (2)
(d) A **software interrupt** is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a *trap* or *exception* and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. For example, if the processor's arithmetic logic unit is commanded to divide a number by zero, this impossible demand will cause a *divide-by-zero exception*. (2)
3. (a) c[:] displays all the elements of the list while c[: -1] displays after reducing one element (i.e. slices the list). (1)
(b) **Object-oriented programming (OOP)** is a programming paradigm that represents concepts as "objects" that have data fields(attributes that describe the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs. Objective-C, Smalltalk, Java and Python are examples of object-oriented programming languages. (2)
(c) A Python decorator is a specific change to the Python syntax that allows us to more conveniently alter functions and methods (and possibly classes in a future version). This supports more readable applications of the DecoratorPattern but also other uses as well.. (2)
(d) A list can store a sequence of objects in a certain order such that you can index into the list, or iterate over the list. List is a mutable type meaning that lists can be modified after they have been created.
Example:
>> stack = [3, 4, 5]
A tuple is similar to a list except it is immutable. There is also a semantic difference between a list and a tuple.
Example:
>>t = 12345, 54321, 'hello'
A dictionary is a key-value store. It is not ordered and it requires that the keys are hashable. It is fast for lookups by key.

Example:

```
>>dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
```

(3)

- (e) Decision Statements or selection statements are the statements which produce a certain output based upon a condition.

Some decision statements commonly supported by all languages are:

(i) **If...Else**

This enables the programmer to provide different paths for the program flow depending on certain conditions. For example: consider a program for dividing two numbers. Division by zero will lead to an error. To avoid this from happening, after obtaining the two numbers from the user the programmer will want to ensure that the denominator is not zero. Hence there needs to be two different program flow options. If the denominator is zero a message saying, "Division not possible" should be displayed. Otherwise the program should carry out the division and display the results. In simpler terms this can be stated as:

If the denominator is zero, then display a message and do not divide
else perform division and display the output.

Syntax:

```
if (condition)
{
    //statements to be executed;
}
else
{
    //statements to be executed;
}
```

(ii) **Nested If**

You can have an 'if' statement within another 'if' statement. This is known as nested 'if'. The program flow will enter into the inner 'if' condition only if the outer 'if' condition is satisfied. In general form nested 'if' will be of the form:

```
if (condition1)
{
    //code to be executed if condition1 is true
    if (condition2)
    {
        //code to be executed if condition2 is true
    }
}
```

(iii) **Switch**

A **switch statement** is a type of selection control mechanism used to allow the value of a variable or expression to change the control flow of program execution via a multiway branch.

Syntax:

In most languages, a switch statement is defined across many individual lines using one or two keywords. A typical syntax is:

- The first line contains the basic keyword, usually switch, case or select, followed by an expression which is often referred to as the *control expression* or *control variable* of the switch statement.
- Subsequent lines define the actual cases (the values) with corresponding sequences of statements that should be executed when a match occurs.

Python supports if...else and nested if.

(3)

4. (a) A **global variable** is a variable that is accessible in every scope. Interaction mechanisms with global variables are called **global environment** mechanisms. The global environment

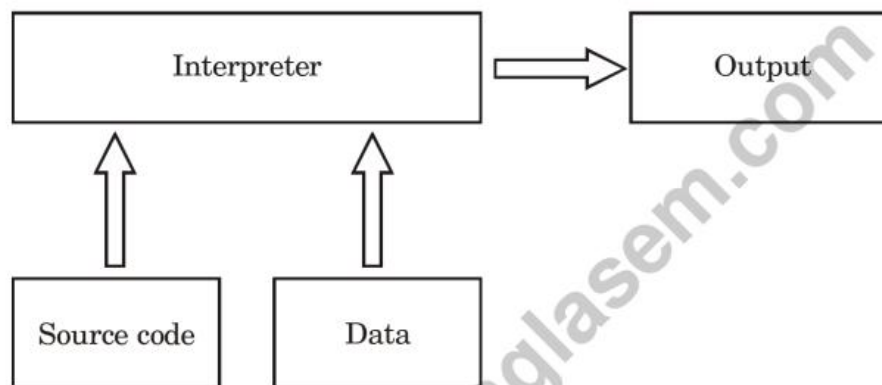
paradigm is contrasted with the local environment paradigm, where all variables are local with no shared memory (and therefore all interactions can be reconducted to message passing).

The scope of global variables is the whole program. (2)

(b) HELLOOSWAL (2)

(c) An **interpreter** is a computer program that executes, i.e. *performs*, instructions written in a programming language. An interpreter generally uses one of the following strategies for program execution:

1. execute the source code directly
2. translate source code into some efficient intermediate representation and immediately execute this
3. explicitly execute stored precompiled code made by a compiler which is part of the interpreter system (1)



(2)

- (d) (i) 1
 (ii) 25
 (iii) False
 (iv) True

(4)

(d) #program to enter a number and print its octal

```
x = 0
while True:
```

(e) try:

```
x = int(raw_input('input a decimal number\t'))
if x in xrange(1,1000000001):
    y = oct(x)
    print y
    z=hex(x)
    print z
    print( zero(x-1)-1)
func xall
xrange()
except ValueError:
    print("Please input an integer, that's not an integer")
    continue
```

(4)

(f) Step1: Start

Step2: Input date, month and year from user.

Step 3: Check if date<=31 and month<=12 and year<=9999

Step4: Print the date in dd/mm/yyyy format and print Yes.

Step5: Else print No

Step6: Stop

(4)

5. (a) 64

(1)

(b) aaabbbcccd

(1)

(c) best stored in a list of dictionaries..

dictionary format: {'number':12453}

(2)

(d) The Corrected Code is:

```
n=[1, 2, 5, 10, 3, 100, 9, 24]
```

```
for e in n :
```

```
if e<5 :
```

```
n.remove(e)
```

```
print n
```

(3)

(e) Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called *exceptions* and are not unconditionally fatal.

It is possible to write programs that handle selected exceptions. Look at the following example, which asks the user for input until a valid integer has been entered, but allows the user to interrupt the program (using Control-C or whatever the operating system supports); note that a user-generated interruption is signalled by raising the **KeyboardInterrupt** exception.

```
>>> while True:
```

```
...     try:
```

```
...         x = int(raw_input("Please enter a number: "))
```

```
...         break
```

```
...     except ValueError:
```

```
...         print "Oops! That was no valid number. Try again..."
```

The **try** statement works as follows.

- First, the *try clause* (the statement(s) between the **try** and **except** keywords) is executed.
- If no exception occurs, the *except clause* is skipped and execution of the **try** statement is finished.
- If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then if its type matches the exception named after the **except** keyword, the except clause is executed, and then execution continues after the **try** statement.
- If an exception occurs which does not match the exception named in the except clause, it is passed on to outer **try** statements; if no handler is found, it is an *unhandled exception* and execution stops with a message as shown above.

(4)

6. (a) `ceil( )` is used to round off a number to next higher integer. `floor( )` rounds off to nearest optimal integer(i.e. if decimal is <5 lower integer else higher)

(1)

(b) First

Fourth

Fifth

(2)

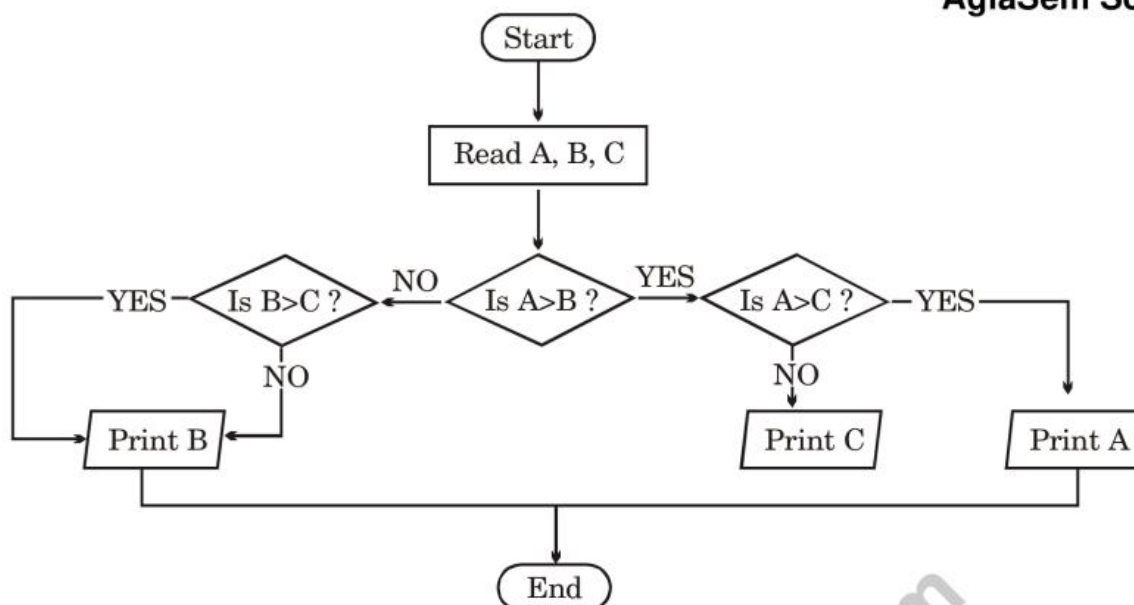
(c) print 'You typed' +line

```
line=line+ 'h'
```

```
print 'You typed the number'+num
```

(3)

(d)



(4)

7. (a) One function performed by School Management System is that it maintains students' attendance records. (1)
- (b) **Bootting** (also known as **booting up**) is the initial set of operations that a computer system performs when electrical power to the CPU is switched on or when the computer is reset. (1)

(c)

Bluetooth	Infrared
1. Not affected by physical inference	Communication stops in presence of interference
2. Range more than infrared	Range less than Bluetooth
3. Bluetooth does not need a direct line of connection	Infrared needs a direct line of connection

(3)

(d) **Third Generation Computers (1964-1971)**

In this era, there were several innovations in various fields of computer technology. These include Integrated Circuits (ICs), Semiconductor Memories, Microprogramming, various patterns of parallel processing and introduction of Operating Systems and time-sharing. In the Integrated Circuit, division there was gradual progress. Firstly, there were small-scale integration (SSI) circuits (having 10 devices per chip), which evolved to medium scale integrated (MSI) circuits (having 100 devices per chip). There were also developments of multi-layered printed circuits.

Parallelism became the trend of the time and there were abundant use of multiple functional units, overlapping CPU and I/O operations and internal parallelism in both the instruction and the data streams. Functional parallelism was first embodied in CDC6600, which contained 10 simultaneously operating functional units and 32 independent memory banks. This device of Seymour Cray had a computation of 1 million flopping point per second (1 M Flops). After 5 years CDC7600, the first vector processor was developed by Cray and it boasted of a speed of 10 M Flops. IBM360/91 was a contemporary device and was twice as fast as CDC6600, whereas IBM360-195 was comparable to CDC7600. In case of language, this era witnessed the development of CPL i.e. combined programming language (1963). CPL had many difficult features and so in order to simplify it Martin Richards developed BCPL - Basic Computer Programming Language (1967). In 1970 Ken Thompson developed yet another simplification of CPL and called it B. (4)

