# 8 Programming Languages

## Introduction

01. Which of the following programming language(s) provides garbage collection automatically
(a) Lisp (b) C++
(c) Fortran (d) C **[ISRO - 2007]**

02. Which of the following is an illegal array definition?
(a) type COLOGNE : (LIME, PINE, MUSK, MENTHOL);
var a : array [COLOGNE] of REAL;
(b) var a : array [REAL] of REAL;
(c) var a : array ['A' .. 'Z'] of REAL;
(d) var a : array [BOOLEAN] of REAL;
**[ISRO - 2008]**

03. Which of the following class of statement usually produces no executable code when compiled?
(a) declaration
(b) assignment statements
(c) input and output statements
(d) structural statements **[ISRO - 2008]**

04. We can make a class abstract by
(a) Declaring it abstract using the virtual keyword
(b) Making at least one member function as virtual function
(c) Making at least one member function as pure virtual function
(d) Making all member function const.
**[ISRO - 2008]**

05. Repeated execution of simple computation may cause compounding of
(a) round-off errors (b) syntax errors
(c) run-time errors (d) logic errors
**[ISRO - 2008]**

06. The feature in object-oriented programming that allows the same operation to be carried out differently, depending on the object, is
(a) Inheritance (b) Polymorphism
(c) Over functioning (d) Overriding
**[ISRO - 2008]**

07. Find the output of the following Java code line
System. out. println(math. floor(−7.4))
(a) – 7 (b) −8
(c) – 7.4 (d) −7.0 **[ISRO - 2013]**

08. What is the right way to declare a copy constructor of a class if the name of the class is Myclass?
(a) MyClass (constant MyClass *arg)
(b) MyClass (constant MyClass &arg)
(c) MyClass (MyClass arg)
(d) MyClass (MyClass *arg) **[ISRO - 2013]**

09. The built in base class in Java, which is used to handle all exceptions is
(a) Raise (b) Exception
(c) Error (d) Throwable
**[ISRO - 2013]**

10. Consider the following Java code fragment. Which of the following statement is true?

```
Line No   Code Statement
1         Public class While
2         {
3            Public void loop ()
4         {
5               int x = 0;
6               While (1)
7               {
8         System.out.println("x plus one is" + (x+1));
9               }
10                 }
11        }
```

(a) There is syntax error in line no.1
(b) There are syntax errors in line nos. 1 & 6
(c) There is syntax error in line no. 8
(d) There is syntax error in line no.6

**[ISRO - 2014]**

11. If only one memory location is to be reserved for a class variable, no matter how many objects are instantiated, then the variable should be declared as _____.
(a) extern (b) static
(c) volatile (d) const

**[ISRO - 2014]**

12. The correct syntax to write "Hi There" in Javascript is
(a) jscript.write ("Hi There")
(b) response.write("Hi There")
(c) print ("Hi There")
(d) print.jscript ("Hi There") **[ISRO - 2015]**

13. If a class C is derived from class B, which is derived from class A, all through public inheritance, then a class C member function can access
(a) Only protected and public data of C and B
(b) Only protected and public data of C
(c) All data of C and private data of A and B
(d) Public and protected data of A and B and all data of C **[ISRO - 2016]**

14. Which of the following operator(s) cannot be overload?
(a) . (Member Access or Dot operator)
(b) ?: (Ternary or Conditional Operator)
(c) :: (Scope Resolution Operator)
(d) All of the above **[ISRO - 2017(May)]**

15. Which of these is a super class of all errors and exceptions in the Java language?
(a) Run Time Exceptions (b) Throwable
(c) Catchable (d) None of the above

**[ISRO - 2017(May)]**

16. Which one of the following are essential features of object oriented language?
A. Abstraction and encapsulation
B. Strictly-typed
C. Type-safe property coupled with sub-type rule
D. Polymorphism in the presence of inheritance
(a) A and B only
(b) A, D and B only
(c) A and D only
(d) A, C and D only

**[ISRO - 2017(Dec)]**

17. A data driven machine is one that executes an instruction if the needed data is available. The physical ordering of the code listing does not dictate the course of execution. Consider the following pseudo-code
(A) Multiply E by 0.5 to get F
(B) Add A and B to get E
(C) Add B with 0.5 to get D
(D) Add E and F to get G
(E) Add A with 10.5 to get C
Assume A, B, C are already assigned values and the desired output is G. Which of the following sequence of execution is valid?
(a) B, C, D, A, E (b) C, B, E, A, D
(c) A, B, C, D, E (d) E, D, C, B, A

**[ISRO - 2018]**

18. Consider the following C++ program

```
int a (int m)
{return ++m ;}
int b(int&m)
{return ++m;}
intc(char &m)
{return ++m;}
void main ()
{
        int p = 0, q =0, r = 0 ;
        p + = a(b(p)) ;
        q + = b(a(q)) ;
        r + = a(c(r)) ;
        cout<<p<<q << r;
}
```

Assuming the required header files are already included, the above program

(a) results in compilation error (b) print 123

(c) print 111 (d) print 322

**[ISRO - 2018]**

| KEY & Detailed Solutions | | | | |
|---|---|---|---|---|
| 01. (a) | 02. (b) | 03. (d) | 04. (c) | 05. (a) |
| 06. (b) | 07. (b) | 08. (b)& (c) | 09. (d) | 10. (d) |
| 11. (b) | 12. (*) | 13. (d) | 14. (d) | 15. (b) |
| 16. (c) | 17. (b) | 18. (a) | | |

**01. Ans: (a)**

**Sol:** C, C++, FORTRAN do not perform automatic garbage collection but are capable of doing it manually.

Lisp is the first high-level programming language that provide automatic garbage collection.

**02. Ans: (b)**

**Sol:** Array index must be integers. Enumerators, characters and boolean can be used in place of an integer but not real.

Options (a) is Enum, (c) is char, (d) is Boolean so it is legal where option (b) is allowing all Real which could be fractional number as well, so its illegal.

**03. Ans: (d)**

**Sol:** while intermediate code generation we lost the structure of the program. So, correct option must be (d).

**04. Ans: (c)**

**Sol:** An abstract class contains at least one pure virtual function. A pure virtual function or abstract function is a virtual function for which we don't have any implementation. A pure virtual function can be declared by using a pure specifier ( = 0 ) in the declaration of a virtual member function in the class declaration.

Option (a), we make virtual function by using virtual keyword. Virtual functions are not enough to make any class abstract. A class with with virtual function declaration without implementation required.

Option (b). That virtual function must not have defined implementation there so its necessary but not sufficient condition

Option (d). Making function constant has no relation with abstract class.

**05. Ans: (a)**

**Sol:** Roundoff error is the difference between an approximation of a number used in computation and its exact (correct) value. In certain types of computation, roundoff error can be magnified as any initial errors are carried through one or more intermediate steps and repeated execution.

So, option (a) is correct.

**06. Ans: (b)**

**Sol:** Polymorphism is the ability of an object to take on many forms. The same entity (function or operator) behaves differently in different scenarios.

Overriding can be implemented in 4 ways Function overloading, Operator overloading, Function overriding, Virtual functions.

Overriding is the part of polymorphism itself so most suitable answer is Polymorphism.

Inheritance is a mechanism in which one object acquires all the properties and behaviors of its parent object.

**07. Ans: (b)**

**Sol:** Floor is a function in java which returns the lower limit. Here –7.4 is a negative number so –8 is lower limit to –7.4 and the higher limit would be –7.

**08. Ans: (b) & (c)**

**Sol:** In this question, a particular language is not mentioned so option (b) and (c) both are correct.

MyClass (constant MyClass &arg) // copy constructor in C++

MyClass (MyClass arg) // copy constructor in Java

**09. Ans: (d)**

**Sol:** Throwable class is the built-in base class used to handle all the exceptions in Java.

Throwable is the base class of all exceptions in java. Exception and Error are sub classes of Throwable.

**10. Ans: (d)**

**Sol:** In Java, while(1) will give compiler time error as it treats as Type mismatch to convert from int to Boolean value. Hence option (d)

Java is case sensitive So, while is a keyword, While is not. While is a valid class name as shown in line 1, so option (a) and (c) can not be the answer.

An equation may be placed in a String operation is valid as shown in line 8, so option (b) is not correct.

**11. Ans: (b)**

**Sol:** Only one copy of each Static data member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.

**12. Ans: None of the options**

**Sol:** Javascript is syntax to print: document.write ("text")

**13. Ans: (d)**

**Sol:** When deriving a class from a public base class, public members of the base class become public members of the derived class and protected members of the base class become protected members of the derived class. A base class's private members are never accessible directly from a derived class, but can be accessed through calls to the public and protected members of the base class. There for class C members can access all of its own class data and Public and protected data of A and B.

| Access | public | protected | private |
|---|---|---|---|
| Same class | yes | yes | yes |
| Derived classes | yes | yes | no |
| Outside classes | yes | no | no |

**14. Ans: (d)**

**Sol:** List of operators that cannot be overloaded

1. Scope resolution operator (::)
2. Pointer-to-member Operator (.*)
3. Member Access or Dot operator (.)
4. Ternary or Conditional Operator (?:)
5. Object size Operator (sizeof)
6. Object type Operator (typeid)

**15. Ans: (b)**

**Sol:** Throwable class is the built-in base class used to handle all the exceptions in Java. Throwable is the base class of all errors and exceptions in java. Exception and Error are sub classes of Throwable.

**16. Ans: (c)**

**Sol:** Abstraction, Encapsulation, Polymorphism and Inheritance are the essential features of a OOPS Language.

(a) Abstraction and Encapsulation is true.
(b) B is false ( Perl is a loosely typed language but supports OOPS),
(c) False
(d) Polymorphism True

**17. Ans: (b)**

**Sol:** Given pseudo-code

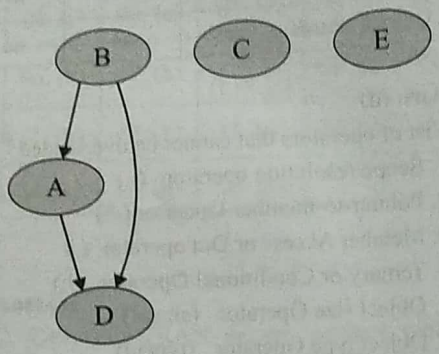
A. F = 0.5*E
B.E = A + B
C.D = 0.5 + B
D.G = E + F

E.C = A + 10.5

It is given that A, B, C are already assigned values. So, instruction A cannot happen before instruction B and instruction D cannot happen before either instruction A or instruction B.

So we can perform B C or E.

Here we can notice that




Option (a) is incorrect as we can't perform instruction D before instruction A.

Option (c) is incorrect as we can't perform instruction A before instruction B.

Option (d) is incorrect as we can't perform instruction D before instruction A and B.

Only option satisfying this is option (b).

**18. Ans: (a)**

**Sol:** 3rd function intc(char &m) will generate compiler error.

Statement q += b(a(q));r += a(c(r)); will also generate compiler error because temporaries can not be bound to non-constant references.

## Basic Arithmetic Operators

01. In C, what is the effect of a negative number in a field width specifier?
(a) the valued are displayed right justified
(b) the valued are displayed centered
(c) the values are displayed left justified
(d) the values are displayed as negative numbers
**[ISRO - 2008]**

02. In Java, after executing the following code what are the values of x, y and z?

```
int x,y = 10, z = 12;
x = y ++ + z++;
```

(a) x = 22, y = 10, z = 12
(b) x = 24, y = 10, z = 12
(c) x = 24, y = 11, z = 13
(d) x = 22, y =11, z = 13
**[ISRO - 2011]**

03. What is the output of the following C program?

```
#include <stdio.h>
#define SQR(x) (x*x)
int main()
{
  int a;
  int b=4;

  a = SQR(b+2)
  printf("%d\n", a);
  return 0;
}
```

(a) 14 (b) 36 (c)18 (d) 20
**[ISRO - 2014]**

04. What is the output of the following C program?

```
#include <stdio.h>
void  main (void)
{
  int shifty;
  shifty = 0570;
  shifty = shifty >> 4;
```

```
shifty = shifty << 6;
printf("The value of shifty is % o \n", shifty);
}
```

(a) The value of shifty is 15c0
(b) The value of shifty is 4300
(c) The value of shifty is 5700
(d) The value of shifty is 2700

**[ISRO - 2014]**

05. The following three 'C' language statements is equivalent to which single statement?

```
y = y+1;
z = x+y;
x = x+1;
```

(a) z = x+y+2;
(b) z = (x++) + (++y);
(c) z = (++x) + (y++);
(d) z = (x++) + (++y) + 1;

**[ISRO - 2014]**

06. What is the output of this C++ program?

```
#include<iostream>
using namespace std;
void square (int *x)
{
  *x = (*x)++ * (*x);
}
void square (int *x, int *y)
{
  *x = (*x) * --(*y);
}
int main ()
{
  int number = 30;
  square (&number, &number);
  count << number;
  return 0;
}
```

(a) 910 (b) 920 (c) 870 (d) 900

**[ISRO - 2017(May)]**

07. Consider the following declaration :

```
structaddr {
        char city[10];
        char street [30];
        int pin ;
};
struct {
        char name [30];
        Int gender ;
        struct addr locate ;
} person, *kd = &person ;
```

Then *(kd → name +2) can be used instead of
(a) person.name +2
(b) kd → (name +2 )
(c) *((*kd).name +2 )
(d) either (a) or (b), but not (c)

**[ISRO - 2018]**

| KEY & Detailed Solutions | | | |
|---|---|---|---|
| 01. (c) | 02. (d) | 03. (a) | 04. (d) |
| 05. (b) | 06. (c) | 07. (c) | |

**01. Ans: (c)**

**Sol: Output:** format specifier with negative number species display the output in spaces which is equivalent to the number and negative make a value left-aligned.

For eg. Format specifier %–20s and %–20d specifies total 20 spaces for the output and –ve number specifies that output will be printed from left hand side(left justified) otherwise it will be right aligned.

```
#include <stdio.h>
int main() {
        printf("\n:%s:","ACE");
        printf("\n:%20s:","ACE");
        printf("\n:%–20s:","ACE");
        printf("\n:%d–20d",100);
        return 0;
}
```

: ACE:
: ACE:
: ACE :
: 100

**02. Ans: (d)**

**Sol:** Y and Z has postfix increment so first current value of y and z will be used to calculate x so x = 10 + 12 = 22 after that X and Y will increment as X = 11 and Y=13.

**03. Ans: (a)**

**Sol:** The #define directives define the identifier as macro, that is instruct the compiler to replace all successive occurrences of identifier with replacement-list
preprocessor expands, a = SQR(b+2) according to its' definition x*x
a = b+2 * b+2 = 4 + 2 * 4+2 = 4+8+2 = 14

**04. Ans: (d)**

**Sol:** shifty = 0570; here shifty is a octal number as it starts with 0.
So 0570 = 000 101 111 000 in binary as each octal digit can be directly converted to 3 bits.
$(0570)_8 = (000\ 101\ 111\ 000)_2$
shifty = shifty >>4; (Right shift by 4 places)
shifty = $(000\ 000\ 010\ 111)_2$
shifty = shifty <<6 ; (Left shift by 6 places:)
shifty = $(010\ 111\ 000\ 000)_2$
$= (2700)_8$

**05. Ans: (b)**

**Sol:** Y is incremented before calculation of z, So y will be preincrement ++y
x incrementing after calculation of z so x is postincrement x++
so z= (x++) + (++y)

**06. Ans: (c)**

**Sol:** Two square functions are defined so here function overloading concept is applied. Both functions have different number of parameters. Main function will call the 2nd square function as it is passing 2 reference So
30* 29 //due to pre-decrement
= 870

**07. Ans: (c)**

**Sol:** *(kd → name +2) will provide the value stored at third character of name. Option a and b will provide the address whereas ,option c will also provide the value at the 3rd index of name.

## Functions

01. The following program
```
main()
{
 inc() ; inc() ; inc() ;
}
inc()
{
  static int x ;
  printf("%d", ++x) ;
}
```
(a) prints 012
(b) prints 123
(c) prints 3 consecutive, but unpredictable numbers
(d) prints 111 **[ISRO - 2015]**

02. Which of the following has the compilation error in C?
(a) int n =17;
(b) char c = 99;
(c) float f = (float) 99.32;
(d) #include<stdio.h> **[ISRO - 2015]**

03. Consider the following statements
#define hypotenuse (a, b) sqrt (a*a + b*b);
The macro call hypotenuse (a+2, b+3);
(a) Finds the hypotenuse of a triangle with sides a+2 and b+3
(b) Finds the square root of $(a+2)^2 + (b+3)^2$
(c) Is invalid
(d) Find the square root of 3 * a + 4 * b + 5 **[ISRO - 2015]**

04. Consider the following C declaration
```
struct {
 short s[5];
 union {
  float y;
  long z;
 }u;
}t;
```
Assume that objects of type short, float and long occupy 2 bytes and 8 bytes, respectively. The meory requirement for variable t, ignoring a alignment considerations, is
(a) 22 bytes (b) 18 bytes
(c) 14 bytes (d) 10 bytes **[ISRO - 2015]**

05. Which one of the following is correct about the statements given below?
I. All function calls are resolved at compile-time in C language
II. All function calls are resolved at compile-time in C++.
(a) Only II is correct
(b) Both I and II are correct
(c) Only I is correct
(d) Both I and II are incorrect **[ISRO - 2016]**

06. What does the following C-statement declare?
int (*f) (int*);
(a) A function that takes an integer pointer as argument and returns an integer
(b) A function that takes an integer as argument and returns an integer pointer
(c) A pointer to a function that takes an integer pointer as argument and returns an integer
(d) A function that takes an integer pointer as argument and returns a function pointer **[ISRO - 2017(May)]**

07. What is the output of the following program?
```
#include <stdio.h>
int tmp=20;
main()
{
  printf("%d",tmp);
  func();
  printf("%d", tmp);
}
```

```
func()
{
  static int tmp = 10;
  printf("%d", tmp);
}
```

(a) 20 10 10 (b) 20 10 20
(c) 20 20 20 (d) 10 10 10

**[ISRO - 2017(May)]**

08. We use malloc and calloc for
(a) Dynamic memory allocation
(b) Static memory allocation
(c) Both dynamic and static memory allocation
(d) None of the above

**[ISRO - 2017(May)]**

09. What is the output of the code given below?

```
#include <stdio.h>
int main()
{
char name [] = "satellites";
int len;
int size;
len = strlen(name);
size = sizeof(name);
printf("%d", len * size);
return 0;
}
```

(a) 100 (b) 110 (c) 40 (d) 44

**[ISRO - 2020]**

10. Following declaration of an array of struct, assumes size of byte, short, int and long are 1,2,3 and 4 respectively. Alignment rule stipulates that n-byte field must be located at an address divisible by n. the fields in a struct are not rearranged, padding is used to ensure alignment. All elements of array should be of same size.

```
Struct complx
Short s
Byte b
Long l
Int i
End complx
Complx C[10]
```

Assuming C is located at an address divisible by 8, what is the total size of C, in bytes?

(a) 150 (b) 160 (c) 200 (d) 240

**[ISRO - 2020]**

| KEY & Detailed Solutions | | | | |
|---|---|---|---|---|
| 01. (b) | 02. (*) | 03. (d) | 04. (b) | 05. (d) |
| 06. (c) | 07. (b) | 08. (a) | 09. (b) | 10. (b) |

**01. Ans: (b)**

**Sol:** Static variable x is not explicitly initialized; so, it would automatically initialised to 0 that too only once and shared by each function calls. inc() is called 3 times so output will be 1,2,3.

**02. Ans: None of above will generate compile time error**

**Sol:** Declare and initialise int variable n to 17
Declare and initialise char variable c to 99. It takes c as input as 99 is ASCII value of 99.
Declare and initialise float variable f = (float)99.32; here 2nd float is used for typecasting.
#include <stdio.h> standard way to include stdio file- No error.

**03. Ans: (d)**

**Sol:** A macro is defined with, hypotenuse (a, b) sqrt (a*a+b*b);
call hypotenuse(a+2,b+3);
hypotenuse = sqrt (a+2*a+2 + b+3*b+3)
= sqrt (3*a + 4*b + 5)

**04. Ans: (b)**

**Sol:** union variable u only creates the memory for only long z' which is the largest size data type inside it = 8 bytes

structure variable t creates the memory for array s and union u = 5*2 + 8 bytes

**05. Ans: (d)**

**Sol:** C does not always support function call resolution at compile time.

Eg 1. logical error can not solve in compile time.

Eg 2. Example of run time resolution of function calls in C.

```
int (*p) ();
scanf("%d", &a);
if(a)
  p = fun1;
else
  p = fun2;
int b = (*p)();
```

C++ Supports "Late Binding". Late Binding is the process of resolving functions related to function calls during Run time.

**06. Ans: (c)**

**Sol:**

| | |
|---|---|
| A function that takes an integer pointer as argument and returns an integer | int f(int*) |
| | |
| A function that takes an integer as argument and returns an integer pointer | int* f (int) |
| | |
| A pointer to a function that takes an integer pointer as argument and returns an integer | int (*f) (int* ); |
| | |
| A function that takes an integer pointer as argument and returns a function pointer | *fp f( int *) |

**07. Ans: (b)**

**Sol:** Main function has no declaration for variable temp so it will print the value of global variable temp. That means whenever printf("%d",tmp); will be executed inside the main it will print value of global temp.

function func has its own local static variable temp so func will use local temp variable instead of global.

**08. Ans: (a)**

**Sol:** To allocate/deallocate memory dynamically in C program, library functions are malloc(), calloc(), realloc() and free() are used. These functions are defined in the <stdlib.h> header file.

**09. Ans: (b)**

**Sol:** char name[ ] = "satellites";

name variable will store all the above char and '\' like shown below

| s | a | t | e | l | l | i | t | e | s | \0 |
|---|---|---|---|---|---|---|---|---|---|---|

The strlen() function calculates the length of a NULL-terminated string. It calculates the length upto null character. (it does not include null char).

len = 10

The sizeof() is a function that is used to calculate the size of its operand. It returns the size of particular variable. That means how many bytes are used to store that variable in the memory. size =11.

You can understand by below condition

```
count = 0; i = 0
while (name[i] ! ='\0')
{ count++; i++;
}
```

Here count variable will provide the length of string and i will provide size of the string as each char takes exact 1 byte.

len = count = 10 size = i = 11

So output would be len * size =10*11=110

**10. Ans: (b)**

**Sol:** Size of complex data type will be 2 + 1 + 4 + 3 = 10 Bytes

But, address is not divisible by 8, so, it needs minimum of 6 bytes of padding.

Now size becomes 16 Bytes (after 6 Bytes of padding)

Total size = 16 × 10 = 160 Bytes

## Control Statements

01. Consider the following psuedocode

```
X : = 1;
i = 1;
while (x ≤ 1000)
begin
   x : = 2x;
   i : = i + 1;
end;
```

What is the value of i at the end of pseudocode?

(a) 4 (b) 5

(c) 6 (d) 7

**[ISRO - 2007]**

02. Consider the following code segment.

```
for (int k = 0; k < 20; k = k + 2)
{
 if (k % 3 = = 1)
 System. out. print (k + " " );
}
```

What is printed as a result of executing the code segment?

(a) 4 16 (b) 4 10 16

(c) 0 6 12 18 (d) 1 4 7 10 13 16 19

**[ISRO - 2008]**

03. Consider the following pseudocode.

```
x : =1;
i ; = 1;
while (x≤500)
begin
      x : = 2^x
      i : = i + 1;
end;
```

What is the value of i at the end of the pseudocode?

(a) 4 (b) 5

(c) 6 (d) 7

**[ISRO - 2011]**

04. What is the output of the following C code?

```
#include <stdio.h>
#include<conio.h>
void main ()
{
  int index;
  for (index=1; index<=5;i++)
  {
        printf("%d", index);
        if(i = = 3)
              continue;
  }
}
```

(a) 1245 (b) 12345 (c) 12245 (d) 12354

**[ISRO - 2011]**

05. In an array of 2N elements that is both 2-ordered and 3-ordered, what is the maximum number of positions that an element can be from its position if the array were 1-ordered?

(a) 1 (b) 2 (c) N/2 (d) 2N-1

**[ISRO - 2013]**

06. What is the output of the following Java program?

```
Class Test
{
  Public static void main (String [] args)
  {
        int x = 0;
        int y = 0;

        for (int z = 0; z < 5; z++)
        {
              if((++x > 2) || (++y >2))
              {
                x++;
              }
        }
        System.out.println(x+" " + y);
  }
}
```

(a) 8 2 (b) 8 5 (c) 8 3 (d) 5 3

**[ISRO - 2013]**

07. Consider the following C code.

```
#include <stdio.h>
#include <math.h>
void main()
{
        double pi = 3.1415926535;
        int a = 1;
        int i;

        for (i=0; i < 3; i++)
         if (a =cos(pi * i/2))
        printf ("%d", 1);
        else
        printf("%d", 0);
}
```

What would the program print?

(a) 0 0 0 (b) 0 1 0 (c) 1 0 1 (d) 1 1 1

**[ISRO - 2013]**

08. How many lines of output does the following C code produce?

```
#include <stdio.h>
float i = 2.0;
float j = 1.0;
float sum = 0.0;
main()
{
        while (i/j > 0.001)
        {
              j + = j;
              sum = sum + (i /j);
              printf("%f\n", sum);
        }
}
```

(a) 8 (b) 9 (c) 10 (d) 11

**[ISRO - 2014]**

09. The output of the following program is

```
main ()
{
  static int x[] = {1, 2, 3, 4, 5, 6, 7, 8};
  int i;
```

```
for (i=2; i<6; ++i)
  x[x[i]] = x[i];
for (i=0; i<8; ++i)
  printf("%d", x[i]);
}
```

(a) 1 2 3 3 5 5 7 8
(b) 1 2 3 4 5 6 7 8
(c) 8 7 6 5 4 3 2 1
(d) 1 2 3 5 4 6 7 8

**[ISRO - 2015]**

10. The for loop

```
for (i = 0; i < 10; ++i)
  printf("%d", i&1);
```

prints

(a) 0101010101
(b) 0111111111
(c) 0000000000
(d) 1111111111

**[ISRO - 2015]**

11. Consider the following segment of C code

```
int j, n;
j = 1;
while (j<=n)
  j = j*2;
```

The number of comparisons made in the execution of the loop for any n > 0 is

(a) $[\log_2 n]*n$
(b) n
(c) $[\log_2 n]$
(d) $[\log_2 n]+1$

**[ISRO - 2016]**

12. What is the output of the following program?

```
main()
{
  int a = 10;
  if ((fork () == 0))
  a++;
  printf("%d\n",a);
}
```

(a) 10 and 11
(b) 10
(c) 11
(d) 11 and 11

**[ISRO - 2017(May)]**

13. What will be the output of the following C code?

```
#include <stdio.h>
main()
{
int i;
for ( i=0; i<5; i++)
{
  int i = 10;
  printf("%d", i);
  i++;
}
return 0;
}
```

(a) 10 11 12 13 14
(b) 10 10 10 10 10
(c) 0 1 2 3 4
(d) Compilation error

**[ISRO - 2017(May)]**

14. What does the following program do when the input is unsigned 16-bit integer?

```
#include <stdio.h>
main()
{
  unsigned int num;
  int i;
  scanf("%u", &num);
  for (i = 0; i < 16; i++)
  {
  printf ("%d", (num << i & 1 << 15) ? 1:0);
  }
}
```

(a) It prints all even bits from num
(b) It prints all odd bits from num
(c) It prints binary equivalent of num
(d) None of the above

**[ISRO - 2017(May)]**

15. Consider the function

```
int func(int num) {
int count = 0;
while(num) {
    count++;
    num >> = 1;
```

```
}
return(count) ;
}
```

For func(435) the value returned is

(a) 9 (b) 8 (c) 0 (d) 10

**[ISRO - 2017(Dec)]**

16. Assume A and B are non-zero positive integers. The following code segment

```
While ( A ! = B ){
  if ( A> B )
    A −= B ;
  else
    B −= A ;
}
cout<<A; // printing the value of A
```

(a) Computes the LCM of two numbers
(b) Divides the larger number by the smaller number
(c) Computes the GCD of two numbers
(d) Finds the smaller of two numbers

**[ISRO - 2018]**

17. Consider the following C code segment :

```
#include <stdio.h>
main()
{
  int i, j, x;
  scanf("%d", &x);
  i = 1 ; j = 1;
  while (i< 10) {
      j = j * i ;
      i = i + 1;
      if ( i == x ) break;
        }
}
```

For the program fragment above, which of the following statements about the variables i and j must be true after execution of this program?
[!(exclamation) sign denotes factorial in the answer ]

(a) $(j = (x - 1)!) \wedge (i \geq x)$
(b) $(j = 9!) \wedge (i = 10)$
(c) $((j = 10!) \wedge (i = 10)) \vee ((j = (x-1)!) \wedge (i = x))$
(d) $((j = 9!) \wedge (i = 10)) \vee ((j = (x-1)!) \wedge (i = x))$

**[ISRO - 2018]**

18. What is the output of the following program?

```
main () {
int x = 2, y =5;
if(x <y) return (x = x+y);
  else printf ("zl");
Printf("z2");
}
```

(a) z2 (b) z1z2
(c) Compilation error (d) None of these

**[ISRO - 2018]**

19. Consider the following C program

```
#include<stdio.h>
main()
{
float sum = 0.0, j = 1.0, i = 2.0;
while ( i/j > 0.001 ) {
        j = j +1;
        sum = sum + i/j ;
        printf("%f\n", sum );
  }
}
```

How many lines of output does this program produce?

(a) 0 – 9 lines of output
(b) 10 – 19 lines of output
(c) 20 – 29 lines of output
(d) More than 29 lines of output

**[ISRO - 2018]**

20. The number of tokens in the following C code segment is

```
switch (inputvalue)
```

{
case 1 : b = c* d; break;
default : b = b ++; break;
}
(a) 27 (b) 29 (c) 26 (d) 24
**[ISRO - 2020]**

21. What is the output of the following 'c' code assuming it runs on a byte addressed little endian machine?

```
#include <stdio.h>
int main()
{
int x ; char *ptr;
x = 622,100,101;
printf("%d", (*(char*)&x) * (x % 3));
return 0;
}
```

(a) 622 (b) 311
(c) 22 (d) 110
**[ISRO - 2020]**

| KEY & Detailed Solutions | | | | |
|---|---|---|---|---|
| 01. (b) | 02. (b) | 03. (b) | 04. (b) | 05. (a) |
| 06. (a) | 07. (c) | 08. (d) | 09. (a) | 10. (a) |
| 11. (d) | 12. (a) | 13. (b) | 14. (c) | 15. (a) |
| 16. (c) | 17. (d) | 18. (d) | 19. (d) | 20. (c) |
| 21. (d) | | | | |

**01. Ans: (b)**

**Sol:** Initialisation: x = 1, i = 1;

| Loop: | x | i |
|---|---|---|
| 1st iteration | $2^1$ | 2 |
| 2nd iteration | $2^2$ | 3 |
| 3rd iteration | $2^4$ | 4 |
| 4th iteration | $2^{16}$ | 5 |
| 5th iteration | Condition Fails as x > 1000 | |

**02. Ans: (b)**

**Sol:** Iteration 1: k = 0 % 3 = 0 condition false nothing printed and k is incremented by 2
Iteration 2: k = 2%3 = 2 condition false nothing printed and k is incremented by 2
Iteration 3 k = 4%3 = 1 condition True so it prints 4 and k is incremented by 2
Iteration 4, k = 6%3 = 0 condition false nothing printed and k is incremented by 2
Iteration 5, k = 8%3 = 2 condition false nothing printed and k is incremented by 2
Iteration 6, k = 10%3 = 1 condition True so it prints 10 and k is incremented by 2
Iteration 7, k = 12%3 = 0 condition false nothing printed and k is incremented by 2
Iteration 8, k = 14%3 = 2 condition false nothing printed and k is incremented by 2
Iteration 9, k = 16%3 = 1 condition True so it prints 16 and k is incremented by 2
Iteration 10, k = 18%3 = 0 condition false nothing printed and k is incremented by 2
Iteration 11, k = 20 loop terminated
Therefore Output is 4 10 16

**03. Ans: (b)**

**Sol:** Initially x=1, i=1
Iteration 1: Condition true and x = 2 and i = 2
Iteration 2: Condition true and x = $2^2$ and i = 3
Iteration 3: Condition true and x = $2^4$ and i = 4
Iteration 4: Condition true and x = $2^{16}$ and i = 5
Iteration 5: Condition False.

**04. Ans: (b)**

**Sol:** When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and next iteration of the loop will begin. In above program there is nothing to skip so it will print 12345

**05. Ans: (a)**

**Sol:** An array is called k-ordered if any one of it's element is at most k places away from its position in the sorted array.

So an array can be termed as 2-ordered array, if it contains an element which is atmost two positions away from its original position in a sorted array.

Array = [0 4 2 5 3 6 8 7 12]

Array = [0 2 3 4 5 6 7 8 12]

[0 1 2 2 1 0 1 1 0]

So in 1-ordered array, an element can be at-most one positions away from its original position in a sorted array.

**06. Ans: (a)**

**Sol:** Question is based on Short-Circuit Evaluation.

Short-Circuit Evaluation - OR operator:

In the case of OR, the expression is evaluated until we get one true result because the result will always be true, independent of the further conditions.

It takes first statement and checks to see if it is true. If the first statement is true, then it returns that object's value without checking the remaining statement. (Short-Circuit)

If the first value is false, only then it checks the second value and then result is based on second value.

z = 0: x = 1 and y = 1, Because x++ will not be executed because condition is false

z = 1: x = 2 and y=2 Because x++; will not be executed because condition is false

z = 2: x=4 and y = 2 , Here x++; will be executed, but ++y will not be executed due to short circuit evaluation

z = 3: x = 6 and y = 2, Here x++ will be executed, but ++y will not be executed due to short circuit evaluation

z = 4: x = 8 and y=2, Here x++ will be executed, but ++y will not be executed due to short circuit evaluation

Hence at the end, It will print 8 2.

**07. Ans: (c)**

**Sol:** 1st iteration: i = 0, a = cos(pi * 0/2) = cos(0) = 1, condition is true print 1

2nd iteration: i = 1, a = cos (pi * 1/2) = 0, condition is true so else part would print 0

3rd iteration: i = 2, a = cos(pi * 2/2) = a = –1, condition is true (as a is non zero) print 1

O/P → 101

**08. Ans: (d)**

**Sol:** Initially i = 2.0 and j = 1.0

i is not changing. So condition i/j < .001 will be false once value of j >2000.

| | | | |
|---|---|---|---|
| 1st iteration | true | print | j=2 |
| 2nd iteration | true | print | j=4 |
| 3rd iteration | true | print | j=8 |

In every iteration value of j is doubled. As we know $2^{10} = 1024$ so $2^{11} = 2048$

That means it will require 11 iteration. At each iteration print statement gets executed so total line will be 11.

**09. Ans: (a)**

**Sol:** We need to focus on

```
for (i = 2; i < 6; ++i)
    x[x[i]] = x[i];
```

1st iteration i= 2, x[ x[2] ] = x[ 2] → x[ 3 ] = 3

2nd iteration i= 3, x[ x[3] ] = x[ 3] → x[ 3 ] = 3

(In previous iteration we modified x[3] )

3rd iteration i= 4, x[ x[4] ] = x[4] → x[ 5 ] = 5

4th iteration i= 5, x[ x[5] ] = x[ 5] → x[ 5 ] = 5

(In previous iteration we modified x[3] )

Next iteration condition fails and array will be array x[ ] = { 1, 2, 3, 3, 5, 5, 7, 8}

**10. Ans: (a)**

**Sol:** Program is applying bitwise & between i and 1

| | |
|---|---|
| 1st iteration | i=0: 0000 AND 0001 = 0000 |
| 2nd iteration | i=1: 0001 AND 0001 = 0001 |
| 3rd iteration | i=2: 0010 AND 0001 = 0000 |
| 4th iteration | i=3: 0011 AND 0001 = 0001 |

..... so on till i =9

It will print 1 for all odd values as the least significant bit (lsb) of odd numbers is 1 and it will generate 1 after & with 1.

**11. Ans: (d)**

**Sol:**

| | | |
|---|---|---|
| For n=1 | j value will be 1,2 | comparisons 2 |
| For n=2 | j value will be 1,2,4 | comparisons 3 |
| For n=3 | j value will be 1,2,4 | comparisons 3 |
| For n=4 | j value will be 1,2,4,8 | comparisons 4 |
| For n=5 | j value will be 1,2,4,8 | comparisons 4 |
| For n=6 | j value will be 1,2,4,8 | comparisons 4 |
| For n=7 | j value will be 1,2,4,8 | comparisons 4 |
| For n=8 | j value will be 1,2,4,8,16 | comparisons 5 |

Correct answer should be $\lfloor \log_2 n \rfloor + 2$ but its not given so closest answer is $\lfloor \log_2 n \rfloor + 1$. They might not be considering comparison of unsuccessful case.

**12. Ans: (a)**

**Sol:** fork() is used to create a new process, which becomes the child process of the caller (It takes no arguments and returns a process ID). After a new child process is created, both processes will execute the next instruction following the fork() system call.

In given question, for parent process fork call will return pid which will make if condition false. Hence parent process will print 10. Child process will execute next instruction i.e. a++ because in child process if-condition is not tested. Execution starts from next instruction. So it will print 11

**13. Ans: (b)**

**Sol:** Here we have two local variables with same name i, so we will refer them as i1 and i2.

**Statement 1:** int i → it is local variable which can be accessed within main function. Lets consider it i1.

**Statement 2:** for(i = 0; i<5; i++) → i1 = 0 and condition will be checked on the value of i1 and i1 will be incremented after each iteration. That means this loop will run total 5 times for i1 value 0,1,2,3,4 and then condition fails.

**Statement 3:** int i = 10 → it is local variable which can be accessed within for loop only. Lets consider it i2.with every iteration it will again initialize to 10.

**Statement 4:** it will print the value of i2 always as it is local to the block of for loop.

**Statement 5**: Increment the value of i2.

Everytime value of i will be initalize to 10 so it will always print 10.

**14. Ans: (c)**

**Sol:** (num<<i&1<<15)?1:0)→((num<<i)&(1<<15)? 1:0));

left shift << has higher precedence than & operator

Here num<<i will make $i^{th}$ bits of number to the MSB

Lets take Num = 24000 → 01011101 11000000

1<<15 → 10000000 00000000

For i=0 Num << 0 → 01011101 11000000

so ((01011101 11000000) & (10000000 00000000) ? 1:0)

print 0

For i=1 Num << 1 → 10111011 10000000

so ((10111011 10000000) & (10000000 00000000) ? 1:0)

print 1

FOR i=2 Num << 2 → 01110111 00000000

Num << 2 → ((01110111 00000000) & (10000000 00000000) ? 1:0)

Print 0

FOR i=3 Num << 3 → 11101110 00000000

Num << 3 → ((11101110 00000000) & (10000000 00000000)? 1:0)

Print 1

In each iteration num binary equivalent is shifting left to i bit.

(num<<i) &(1<<15) results depend only upon the MSB of num <<i

That means program prints MSB of (num<<i)

This will be binary equivalent of num itself.

15. **Ans: (a)**

**Sol:** num >> = 1 means num = num>>1 (divide the number by 2)

right shift operator(>>)is used to shift the number one position to right, the output value will be exactly number / 2.

In above program count variable is simply counting how many times loops statements executed and at each executions number is divided by 2,

we get: 435, 217, 108, 54, 27, 13, 6, 3, 1.

Therefore, the count is 9.

16. **Ans: (c)**

**Sol:** This is Euclidean algo for determining HCF/GCD of two numbers which works in this way :

The Algorithm

The Euclidean Algorithm for finding GCD(A,B) is as follows:

If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and we can stop.

If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and we can stop.

Write A in quotient remainder form (A = B·Q + R)

Find GCD(B,R) using the Euclidean Algorithm since GCD(A,B) = GCD(B,R)

If we subtract smaller number from larger (we reduce larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.

17. **Ans: (d)**

**Sol:** Let's solve the above code first.

iteration 1st i=1 ; j =1

iteration 2nd i=2 ; j=2

iteration 3rd i=3 ; j=6

iteration 4th i=4 ; j=24

iteration 5th i=5 ; j=120

The given while loop terminates

case 1) j will be executed upto i = x–1 and at i = x condition false then j will be j= (x–1)! For example in the above example if x=5 then at i=5 condition will be false and j=24.

It gives (( j = (x – 1)!) ∧ (i = x ))

case 2 j will be executed upto i = 9 and at i=10 condition false so j=9!

It gives (( j = 9!) ∧ (i = 10))

Final output

(( j = (x – 1)!) ∧ (i = x )) ∨ (( j = 9!) ∧ (i = 10))

(( j = 9!) ∧ (i = 10)) ∨ (( j = (x – 1)!) ∧ (i = x ))

18. **Ans: (d)**

**Sol:** x<y is true so x = 2 + 5 = 7 and return x will be executed and programs completes.

No print statement will be executed.

19. **Ans: (d)**

**Sol:** Initially i = 2.0 and j = 1.0

i is not changing. So condition i/j <.001 will be false once value of j >2000.

1st iteration i/j=2 i= 2 j=2 print→sum

2nd iteration i/j=1 i= 2 j=2 print→sum

3rd iteration i/j=1 i= 2 j=4 print→sum

In every iteration value of j is doubled.

Condition will be false when j>2000.

As we know $2^{10} = 1024$ so $2^{11} = 2048$

That means it will require 2001 iterations. At each iteration print statement gets executed so total line will be more than 29 lines of output.

20. **Ans: (c)**

**Sol:**

| S.No. | Token |
|---|---|
| 1 | switch |
| 2 | ( |
| 3 | inputvalue |
| 4 | ) |
| 5 | { |
| 6 | case |
| 7 | 1 |
| 8 | : |
| 9 | b |

| | |
|---|---|
| 10 | = |
| 11 | c |
| 12 | * |
| 13 | d |
| 14 | ; |
| 15 | break |
| 16 | ; |
| 17 | default |
| 18 | : |
| 19 | b |
| 20 | = |
| 21 | b |
| 22 | ++ |
| 23 | ; |
| 24 | break |
| 25 | ; |
| 26 | } |

**21. Ans: (d)**

**Sol:** Assignment operator (=) has higher priority than comma operator so x will store 622.

Big Endian representation: The most significant byte (the "big end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.

Little Endian representation: The least significant byte (the "little end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next three bytes in memory.

Little Endian

| | | |
|---|---|---|
| 622 in binary | 000000 10<br>MSB | 01101110<br>LSB |

Lets solve (*(char *)&x) * (x % 3)

Here x % 3 = 622 % 3 = 1

(*(char *)&x) → accessing only 1 byte of variable x and its little-endian so byte will be LSB 01101110 whose decimal value is 110

So (*(char *)&x) * (x % 3) → 110 *1 =110

01. Study the following program:

```
// precondition : x >= 0
public void demo(int x)
{
  System.out.print(x % 10);
  if(x / 10) ! = 0)
  {
        demo(x / 10);
  }
  System.out.print (x % 10);
}
```

Which of the following is printed as a result of the call demo (1234)?

(a) 1441 (b) 3443

(c) 12344321 (d) 43211234

**[ISRO - 2007]**

02. What is the value of F(4) using the following procedure:

```
function F (K : integer): integer;
  begin
if(k<3) then F: = K else F: = F(k–1)*F(K–2) +
F (k–3) end;
```

(a) 5 (b) 6 (c) 7 (d) 8

**[ISRO - 2008]**

03. Consider the following C function:

```
int f(int n)
{
static int i = 1; if (n> = 5) return n;
n = n + i; i++;
return f(n);
}
```

The value returned by f(1) is

(a) 5 (b) 6

(c) 7 (d) 8

**[ISRO - 2008]**

04. What is the time complexity for the following C module?
Assume that n > 0;

```
int module(int n)
{
    if(n ==1)
      return 1;
    else
      return (n + module(n-1));
}
```

(a) O(n) (b) O(log n)
(c) O($n^2$) (d) O(n!) **[ISRO - 2014]**

05. Consider the function

```
int fun(x: integer)
{
    If x > 100 then fun = x – 10;
    else
    fun = fun(fun(x+11));
}
```

For the input x = 95, the function will return
(a) 89 (b) 90 (c) 91 (d) 92 **[ISRO - 2017(Dec)]**

06. In multi-programmed systems, it is advantageous if some programs such as editors and compilers can be shared by several users.
Which of the following must be true of multi-programmed systems in order that a single copy of a program can be shared by several users?
I. The program is a macro
II. The program is recursive
III. The program is reentrant
(a) I only (b) II only
(c) III only (d) I, II and III **[ISRO - 2018]**

07. Let P be a procedure that for some inputs calls itself (i.e. is recursive). If P is guaranteed to terminate, which of the following statement(s) must be true?
I. P has a local variable
II. P has an execution path where it does not call itself
III. P either refers to a global variable or has at least one parameter
(a) I only (b) II only
(c) III only (d) II and III **[ISRO - 2018]**

08. Consider the following C code segment

```
int f ( int x )
{
    if ( x<1 ) return 1;
    else return ( f(x–1) + g(x) ) ;
}
int g ( int x )
{
    if ( x<2 ) return 2 ;
    else return ( f(x–1) + g(x/2));
}
```

Of the following, which best describes the growth of *f(x)* as a function of x ?
(a) Linear (b) Exponential
(c) Quadratic (d) Cubic **[ISRO - 2018]**

09. Consider the following recursive C function that takes two arguments

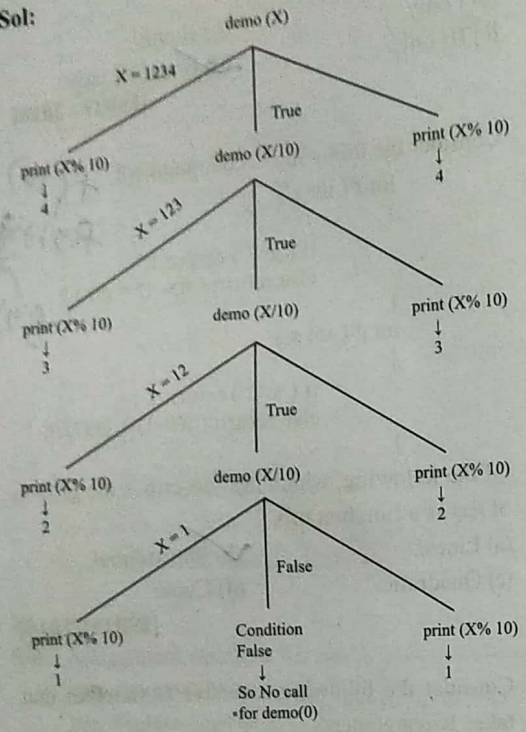
```
unsigned int rer (unsigned int n, unsigned int r) {
if (n > 0) return (n%r + rer (n/r, r)) ;
else return 0;
}
```

What is the return value of the function rer when it is called as rer (513, 2)?
(a) 9 (b) 8 (c) 5 (d) 2 **[ISRO - 2020]**

| KEY & Detailed Solutions | | | | |
|---|---|---|---|---|
| 01. (d) | 02. (a) | 03. (c) | 04. (a) | 05. (c) |
| 06. (c) | 07. (d) | 08. (b) | 09. (d) | |

**01. Ans: (d)**

**Sol:**




**02. Ans: (a)**

**Sol:** F(0) = 0

F(1) = 1

F(2) = 2

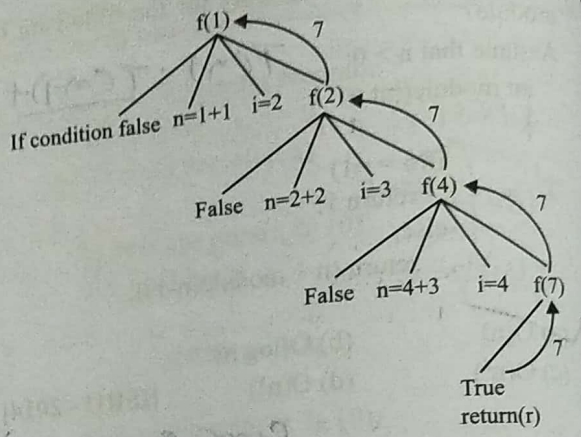
F(3) = F(2)*F(1) + F(0) = 2

F(4) = F(3)*F(2) + F(1) = 5

**03. Ans: (c)**

**Sol:** Variable i is static so its will be initialized only once at the load time. In the run time statement static int i = 1; will never be executed.

Static int i = 1 2 3 4




**04. Ans: (a)**

**Sol:** Recurrence relation T(n) = T(n−1)+c

= T(n−2) + c + c =T(n−2)+2c

= T(n−3) +3c

.....

.....

= T(n−n+1) + (n−1)*c

= T(1) + (n−1)*c

= O(n)

**05. Ans: (c)**

**Sol:** fun(95) = fun(fun(106)) = fun(96) = fun(fun(107)) =fun(97) =fun(fun(108)) =fun(98) = fun(fun(109)) =fun(99) = fun(fun(110))=fun(100)=fun(fun(111)) = fun(101) = 91

**06. Ans: (c)**

**Sol:** Macros are used to provide a program generation facility through macro expansion. A macro is defined with the pre-processor directive. Macros are pre-processed which means that all the macros would be processed before your program compiles. Macros are used to make a sequence of computing instructions available to the programmer as a single program statement, making the programming task less tedious and less error-prone.

Recursive describes a function or method that repeatedly calculates a smaller part of itself to arrive at the final result.

Reentrant code is commonly required in operating systems and in applications intended to be shared in multi-user systems.

A programmer writes a reentrant program by making sure that no instructions modify the contents of variable values in other instructions within the program. Each time the program is entered for a user, a data area is obtained which keeps all the variable values for that user.

The data area is in another part of the memory from the program itself.

When the program is interrupted to give another user a turn to use the program, information about the data area associated with that user is saved.

When the interrupted user of the program is once again given control of the program, information in the saved data area is recovered and the program can be reentered without concern that the previous user has changed some instruction within the program.

**07. Ans: (d)**

**Sol:** If P is guaranteed to terminate that means a base condition exist where function is not calling itself. So, statement (II) is True.

At each call function p will create a new copy of local variables in the stack storage area. With the new copy of local variable base condition can not be satisfied as Local variable will always initialized and compute the same value at every function call. Function P can terminate if base condition is either dependent on a global variable or base condition deepened upon a parameter which is passes as function argument. S Statement III is True.

**08. Ans: (b)**

**Sol:** Recursive relation for function g(int x)

$$g(x) = f(x-1) + g(x/2)$$

Recursive relation for function f(int x)

$f(x) = f(x-1) + g(x)$

$f(x) = f(x-1) + f(x-1) + g(x/2) \rightarrow 2*f(x-1) + g(x/2)$

$f(x) > 2.f(x-1)$

$f(x) > 2f(x-1)$

$f(x) > 2^2 f(x-2)$

$f(x) > 2^3 f(x-3)$

$f(x) > 2^4 f(x-4)$

.....

.....

$f(x) > 2^{x-1} f(1)$

.....

.....

$\Rightarrow f(x) > (2^x) f(1)$

it's exponential. so option (B)

**09. Ans: (d)**

**Sol:**

| | |
|---|---|
| 1st call to function rer | n=513 so condition True rer(513,2)=1+rer(256,2) |
| 2nd call to function rer | n=256 so condition True rer(256,2)=0+rer(128,2) |
| 3rd call to function rer | n=128 so condition True rer(128,2)=0+rer(64,2) |
| 4th call to function rer | n=64 so condition True rer(64,2)=0+rer(32,2) |
| 5th call to function rer | n=32 so condition True rer(32,2)=0+rer(16,2) |
| 6th call to function rer | n=16 so condition True rer(16,2)=0+rer(8,2) |
| 7th call to function rer | n=8 so condition True rer(8,2)=0+rer(4,2) |
| 8th call to function rer | n=4 so condition True rer(4,2)=0+rer(2,2) |
| 9th call to function rer | n=2 so condition True rer(2,2)=0+rer(1,2) |
| 10th call to function rer | n=1 so condition True rer(1,2)=1+rer(0,2) |
| 11th call to function rer | rer(0,2)=return 0 |
| Output will be 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 = 2 | |

## Pointers

01. Which of the following is TRUE with respect to a Reference?
(a) A reference can never be NULL
(b) A reference needs an explicit dereferencing mechanism
(c) A reference can be reassigned after it is established
(d) A reference and pointer are synonymous
**[ISRO - 2014]**

02. What is the output of this C Code?
```
#include <stdio.h>
        void main()
{
  int k = 5;
  int *p = &k;
  int **m = &p;
  printf("%d%d%d\n", k, *p, **m);
}
```
(a) 5 5 5 (b) 5 5 junk
(c) 5 junk junk (d) Compile time error
**[ISRO - 2016]**

03. What will be output of following program? Assume that you are running this program in little-endian processor.
```
#include<stdio.h>
int main() {
short a = 320;
char *ptr;
  ptr = (char*)&a;
  printf("%d", *ptr);
return 0;
}
```
(a) 1 (b) 320
(c) 64 (d) Compilation Error
**[ISRO - 2016]**

04. Consider the following C function
```
#include <stdio.h>
  int main(void)
{
  char c[] = "ICRBCSIT17";
  char *p=c;
  printf("%s", c+2[p] – 6[p]–1);
  return 0;
}
```
The output of the program is
(a) SI (b) IT
(c) T1 (d) 17
**[ISRO - 2017(Dec)]**

05. Consider the following program
```
main()
{
int x = 1;
printf ("%d", (*char(char *)&x)) ;
}
```
Assuming required header files are included and if the machine in which this program is executed is little-endian, then the output will be
(a) 0 (b) 99999999
(c) 1 (d) unpredictable
**[ISRO - 2018]**

06. What is the output in a 32 bit machine with 32 bit compiler?
```
#include<stdio.h>
rer(int **ptr2,int **ptr1)
{
  int* ii;
  ii = *ptr2;
  *ptr2 =* ptr1;
  *ptr1=ii;
  **ptr1* = **ptr2;
  **ptr2 + = **ptr1;
}
```

```
void main ()
{
  int var1=5, var2 =10;
  int *ptr1=&var1, *ptr2=&var2;
  rer(&prt1, &ptr2);
  printf("%d %d", var2,var1);
}
```

(a) 60 70 (b) 50 50
(c) 50 60 (d) 60 50

**[ISRO - 2020]**

| KEY & Detailed Solutions | | |
|---|---|---|
| 01. (a) | 02. (a) | 03. (c) |
| 04. (d) | 05. (*) | 06. (d) |

**01. Ans: (a)**

**Sol:** References cannot be null, whereas pointers can; every reference refers to some object, although it may or may not be valid. A reference can never be re-assigned once it is established.

So, option (a) is correct.

(a) A reference can never be NULL. True
References cannot be null,every reference refers to some object, although it may or may not be valid. So its true.

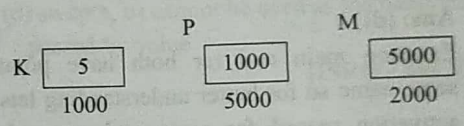
(b) Reference need an explicit dereferencing mechanism False.
A reference variable is an alias, that is, another name for an already existing variable. Reference does not need an explicit dereferencing mechanism

(c) A reference can be reassigned after it is established False
A reference once established cannot be changed so Once a reference is created, it cannot be later made to reference another object.

(d) A reference and pointer are synonymous. False | They are entirely different concept as given below.
A pointer is a variable that holds memory address of another variable. A pointer needs to be dereferenced with * operator to access the memory location it points to.
A reference variable is an alias, that is, another name for an already existing variable. A reference, like a pointer, is also implemented by storing the address of an object.

**02. Ans: (a)**

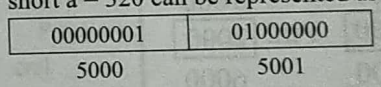
**Sol:**




K → 5
P → 1000 * P → 5
M → 5000 * M→1000 **M→5

**03. Ans: (c)**

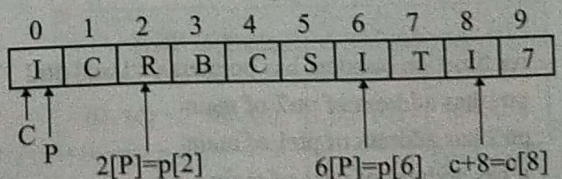
**Sol:** short a = 320 can be represented as




Little-endian →The least significant byte (LSB) value, is at the lowest address.
ptr = (char *)&a;
short variable a is typecasted to char so ptr will take address of least signficant byte (8 bits of right hand side) so ptr = 5000
*ptr = *(5001) = 64

**04. Ans: (d)**

**Sol:** String c = "ICRBCSIT17"



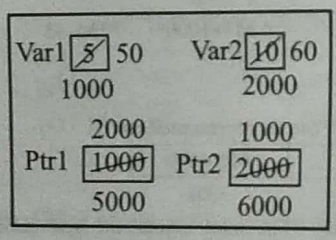
arr[i]= *(arr+i) = *(i+arr) = i[arr]
suppose string c starting address is 2000 so
p=c=2000
*(p+i)=c[i]=i[c]
2[p] = p[2] ='R'
6[p] = p[6] ='I'
printf("%s", c+2[p] – 6[p] – 1);
printf("%s", c+ 'R' – 'I' – 1);
printf("%s", c+8);
So output will be 17

**05. Ans: None of the above**

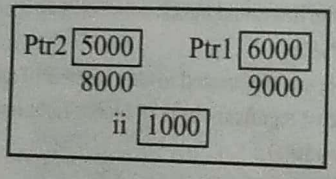
**Sol:** (*char(char*)&x)) will generate error.

**06. Ans: (d)**

**Sol:** Function main and rer both have pointers of same name so for better understanding lets create activation record for main and rer and create memory for each variable.




Main activation record




rer activation record

Main function has pointers ptr1 and ptr2
ptr1 has address of var1
ptr2 has address of var2

rer function pointers has pointers ptr1 and ptr2
ptr1 has address of ptr2 of main
ptr2 has address of ptr1 of main
ii has address of var1 of main

now *ptr2 = *ptr1will directly update the ptr1 of main and now it has address of var2.
now *ptr1 = ii will directly update the ptr2 of main and now it has address of var1.

**ptr1 * = **ptr2; → will update the value of var1 of main to 5*10 = 50

**ptr2 += **ptr1; → will update the value of var2 of main to 50 + 10 = 60

Once function rer is complete then statement printf("%d %d ", var2, var1); of main will be executed and output will be 60, 50.

## Parameter Passing Mechanisms

01. Consider the following pseudocode:

```
x : integer : =1
y : integer = 2
procedure add
  x : = x + y
procedure second (P: procedure)
  x : integer : =2
  P()
procedure first
  y : integer : = 3
  second(add)
first()
write _integer(x)
```

What does it print if the language uses dynamic Scoping with deep binding?

(a) 2 (b) 3 (c) 4 (d) 5

**[ISRO - 2013]**

02. Consider the following code segment

```
void foo(int x, int y)
 {
   x+=y;
   y+=x;
 }
main()
 {
     int x =5.5;
     foo(x,x);
 }
```

What is the final value of x in both call by value and call by reference, respectively?

(a) 5 and 16
(b) 5 and 12
(c) 5 and 20
(d) 12 and 20

**[ISRO - 2015]**

03. Consider the following C function

```
void swap ( int x, int y)
{
int tmp;
tmp =x;
x = y;
y = tmp;
}
```

In order to exchange the values of two variables a and b:

(a) Call swap (a, b)
(b) Call swap (&a, &b)
(c) swap(a, b) cannot be used as it does not return any value
(d) swap(a, b) cannot be used as the parameters passed by value

**[ISRO - 2017(May)]**

04. In the following procedure

```
Integer procedure P(X,Y);
Integer X, Y;
value x;
begin
  K = 5;
  L = 8;
  P = x + y;
end
```

X is called by value and Y is called by name. If the procedure were invoked by the following program fragment

```
K = 0;
L = 0;
Z = P(K, L);
```

then the value Z would be set equal to

(a) 5 (b) 8 (c)13 (d) 0

**[ISRO - 2020]**

| KEY & Detailed Solutions | | | |
|---|---|---|---|
| 01. (c) | 02. (c) | 03. (d) | 04. (b) |

**01. Ans: (c)**

**Sol:** Here, execution starts with statement first()
It will call the procedure first
Local variable "y" is created with value 3
Now second() is called and function "add" is passed as an argument to it.
Due to deep-binding, the scope of variables in "add" gets assigned here i.e. add function will take value of y from calling function (first) and value of x from global variable as local variable of second has not been created yet.
Now, control goes to "second".
Local variable "x" is created with value 2
function p (which is "add") is called.
Now control goes to Add function
due to deep binding, x is binded to the "global x".
So, global x = 1 + 3 = 4
At the end write_integer (x) will be executed and value of global variable x will be printed.

**02. Ans: (c)**

**Sol:** X is an integer variable so value of 5.5 gets truncated and only the integer part 5 will be stored.
For call be by value, changes done by function foo will not be reflected in the local variable of main so value will remain 5.
for call by reference, function foo variables x and y are pointing to the same variable.
x = 5 + 5 = 10. It will be reflected in the main variable local variable as well and both x and y both refer to 10.
y = 10 + 10 = 20. It will be reflected in the main's local variable as well.

**03. Ans: (d)**

**Sol:** In order to exchange the values of two variables a and b of the calling function need to pass the address of variable a and b i.e. &a and &b and that should be accepted by the called function parameter which must be pointer type.
But in given swap function declaration parameters are passed by values so swap(a,b) cannot be used as the parameters are passed by value

**04. Ans: (b)**

**Sol:** Call by name refers to a parameter passing technique in which formal parameter that is present in the function definition is replaced by the actual parameter from the caller function. So in above program Y will be replaced by L.
Call by value is a technique in which variable values from actual parameters are copied in the variables of formal parameters.
Initially k= 0 and L = 0;
When we execute the statement Z=P(K,L) means we are calling Integer procedure P(X,Y); here value of x will be 0 as it is call by value but Y will be replaced by L as it is call by name.
Now in the function we are executing K = 5; L = 8 then P = x + y; here value of x is 0 but y is 8 as y is nothing but L due to call by name so P = 0 + 8;
So it will return 8 and Z = P(K,L) = 8.