**Dr. D. Y. Patil Institute of Technology, Pimpri, Pune**

**Department of Computer Engineering**

**Subject-PPL(210255)**

**MCQ- Unit-2 Structuring the Data, Computations and Program**

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is not valid variable name? |
| ((OPTION\_ A)) | int $index; |
| ((OPTION\_ B)) | int index1; |
| ((OPTION\_ C)) | int ind1x; |
| ((OPTION\_ D)) | int index; |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | A |
| ((EXPLANA TION)) (OPTIONAL  ) | The variable name should not begin with special symbol like @, #, $, % and so on |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is true for variable names in C |
| ((OPTION\_ A)) | They can contain alphanumeric characters as well as special characters |
| ((OPTION\_ B)) | Variable names cannot start with a digit |
| ((OPTION\_ C)) | It is not an error to declare a variable to be one of the keywords (like goto, static) |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is true of I-value and r-values? |
| ((OPTION\_ A)) | An I-value is a logical value and an r-value is a real value |
| ((OPTION\_ B)) | I-Value are always to the left of r-value |
| ((OPTION\_ C)) | An I-value refers to a variable location while an r-value to its current value |
| ((OPTION\_ D)) | I-value are local and r-value are relative |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | What will be the output of the following Java program?  class Output  {  public static void main(String args[])  {  final int a=10,b=20;  while(a<b)  {    System.out.println("Hello");  }  System.out.println("World");    }  } |
| ((OPTION\_ A)) | Hello |
| ((OPTION\_ B)) | run time error |
| ((OPTION\_ C)) | Hello world |
| ((OPTION\_ D)) | compile time error |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | D |
| ((EXPLANA TION)) (OPTIONAL  ) | Every final variable is compile time constant. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | What will be the output of the following Java program?  class comma\_operator  {  public static void main(String args[])  {  int sum = 0;  for (int i = 0, j = 0; i < 5 & j < 5; ++i, j = i + 1)  sum += i;  System.out.println(sum);  }  } |
| ((OPTION\_ A)) | 5 |
| ((OPTION\_ B)) | 6 |
| ((OPTION\_ C)) | 14 |
| ((OPTION\_ D)) | Compilation error |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) | Using comma operator, we can include more than one statement in the initialization and iteration portion of the for loop. Therefore both ++i and j = i + 1 is executed i gets the value – 0,1,2,3,4 & j gets the values -0,1,2,3,4,5. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of this statement is incorrect? |
| ((OPTION\_ A)) | switch statement is more efficient than a set of nested ifs |
| ((OPTION\_ B)) | two case constants in the same switch can have identical values |
| ((OPTION\_ C)) | switch statement can only test for equality, whereas if statement can evaluate any type of boolean expression |
| ((OPTION\_ D)) | it is possible to create a nested switch statements |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) | No two case constants in the same switch can have identical values. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which is the longest scope in the following code?  #include<stdio.h>  int x;  int main()  {  int y;  fun();  return 0;  }  Void fun()  {  int z;  } |
| ((OPTION\_ A)) | x |
| ((OPTION\_ B)) | y |
| ((OPTION\_ C)) | z |
| ((OPTION\_ D)) | Both a & b |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | A |
| ((EXPLANA TION)) (OPTIONAL  ) | The variable is a global variable and its scope is entire program including all the function |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The variable which can be accessed by all modules in a program , are called -------- |
| ((OPTION\_ A)) | Local variables |
| ((OPTION\_ B)) | Internal variables |
| ((OPTION\_ C)) | External variable |
| ((OPTION\_ D)) | Global variables |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | D |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The value of an automatic variable that is declared but not initialized will be ------------- |
| ((OPTION\_ A)) | 0 |
| ((OPTION\_ B)) | 1 |
| ((OPTION\_ C)) | Unpredictable |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) | If the variable is not initialized then it takes the garbage value. Hence the unpredictable value is present in the uninitialized variable |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is not data type in Pascal? |
| ((OPTION\_ A)) | Real |
| ((OPTION\_ B)) | Float |
| ((OPTION\_ C)) | Char |
| ((OPTION\_ D)) | Integer |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | What will happen if null pointer is converted to bool----- |
| ((OPTION\_ A)) | The bool value is evaluated to true |
| ((OPTION\_ B)) | The bool value is evaluated to false |
| ((OPTION\_ C)) | Error is raised |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | In pointer, the nonzero pointer is converted to true and zero pointer is converted to false |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | int main(void)  {  Char name=\*p;  ‘p’=10 /\* assigning to character P number 10\*/  Return 0;  } |
| ((OPTION\_ A)) | Name will contain value 10 |
| ((OPTION\_ B)) | P will contain 10 |
| ((OPTION\_ C)) | Syntax error |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) | The assignment ‘p’=10 will cause the error; I-value required as left operand of assignment |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Choose the correct statement |
| ((OPTION\_ A)) | Reference is stored on stack |
| ((OPTION\_ B)) | Reference is stored on heap |
| ((OPTION\_ C)) | Reference is stored on queue |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | A |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Discriminated unions are supported by----- |
| ((OPTION\_ A)) | C |
| ((OPTION\_ B)) | C++ |
| ((OPTION\_ C)) | ML |
| ((OPTION\_ D)) | All of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | In C, reference is declared using symbol----- |
| ((OPTION\_ A)) | \* |
| ((OPTION\_ B)) | & |
| ((OPTION\_ C)) | && |
| ((OPTION\_ D)) | ! |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Chose the correct statement   1. Array of references can be created 2. Change in reference changes the referent |
| ((OPTION\_ A)) | Only I is correct |
| ((OPTION\_ B)) | Only II is correct |
| ((OPTION\_ C)) | Both I and II is correct |
| ((OPTION\_ D)) | Neither I and II is correct |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Choose the correct statement |
| ((OPTION\_ A)) | Reference must be initialized within a function |
| ((OPTION\_ B)) | Reference must be initialized outside a function |
| ((OPTION\_ C)) | Reference must be always initialized |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Choose the correct statement |
| ((OPTION\_ A)) | A reference is not a constant pointer |
| ((OPTION\_ B)) | Reference is automatically de-referenced |
| ((OPTION\_ C)) | Both a & b |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Choose the correct statement |
| ((OPTION\_ A)) | A reference variable need not be de-referenced to access value |
| ((OPTION\_ B)) | A reference variable need to be de-referenced to access value |
| ((OPTION\_ C)) | It depends upon the type of reference whether to de-refer it to access value |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE)) | A |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Choose the correct statement  I The variable and its reference are linked together  II We can change the value of variable via its reference |
| ((OPTION\_ A)) | Only I |
| ((OPTION\_ B)) | Only II |
| ((OPTION\_ C)) | Both I & II |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | If an expression contain double, float, and long then the data type of that expression becomes----- |
| ((OPTION\_ A)) | Int |
| ((OPTION\_ B)) | Float |
| ((OPTION\_ C)) | Long |
| ((OPTION\_ D)) | Double |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | d |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Explicit type conversion is known as ------ |
| ((OPTION\_ A)) | Conversion |
| ((OPTION\_ B)) | Casting |
| ((OPTION\_ C)) | Separation |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | What is strong type system? |
| ((OPTION\_ A)) | The type system in which only built-in data types are allowed |
| ((OPTION\_ B)) | The type system in which only user defined data types are allowed |
| ((OPTION\_ C)) | The type system that guarantees not to generate type errors |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | For representing logical values ----- data type is used in C++ |
| ((OPTION\_ A)) | int |
| ((OPTION\_ B)) | bool |
| ((OPTION\_ C)) | char |
| ((OPTION\_ D)) | All of the above |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) | The C++ allows the boolen data type for denoting true or false values |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Looping in a program means----- |
| ((OPTION\_ A)) | Jumping to the specified branch of program |
| ((OPTION\_ B)) | Repeat the specified lines of code |
| ((OPTION\_ C)) | Both of above |
| ((OPTION\_ D)) | None of these |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | What is required to reference an element in an array? |
| ((OPTION\_ A)) | Array name |
| ((OPTION\_ B)) | Index value of the element |
| ((OPTION\_ C)) | Element value |
| ((OPTION\_ D)) | Both a and b |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | D |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The following the exit controlled loop------------- |
| ((OPTION\_ A)) | While |
| ((OPTION\_ B)) | do-while |
| ((OPTION\_ C)) | for |
| ((OPTION\_ D)) | go to |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | B |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The scope of a variable refer to----- |
| ((OPTION\_ A)) | The length of the variable |
| ((OPTION\_ B)) | The name of the variable |
| ((OPTION\_ C)) | The accessibility of the variable |
| ((OPTION\_ D)) | The lifetime of the variable |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The goal of structured programming is to--- |
| ((OPTION\_ A)) | Have well indented programs |
| ((OPTION\_ B)) | Be able to infer the flow of control from the compiled code |
| ((OPTION\_ C)) | Be able to infer the flow of control from the program text |
| ((OPTION\_ D)) | Avoid the use of GOTO statements |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Functions and subprograms are both same |
| ((OPTION\_ A)) | True |
| ((OPTION\_ B)) | False |
| ((OPTION\_ C)) |  |
| ((OPTION\_ D)) |  |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | b |
| ((EXPLANA TION)) (OPTIONAL  ) | A subprogram consists of procedures and functions. Both of them are collectively called subprograms. So, subprogram is not same as a function but a function is a part of subprogram in case of VHDL. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | The function is called from the \_\_\_\_\_\_\_\_ |
| ((OPTION\_ A)) | Function itself |
| ((OPTION\_ B)) | Library |
| ((OPTION\_ C)) | Main code |
| ((OPTION\_ D)) | Package |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | c |
| ((EXPLANA TION)) (OPTIONAL  ) | The function which is once declared is always called from the main code. Whenever a function call occurs, the control is passed to the space where the function is defined. Then, the function is executed till a RETURN statement comes, which returns the control to main code. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which among the following best defines abstraction? |
| ((OPTION\_ A)) | Hiding the implementation |
| ((OPTION\_ B)) | Showing the important data |
| ((OPTION\_ C)) | Hiding the important data |
| ((OPTION\_ D)) | Hiding the implementation and showing only the features |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | d |
| ((EXPLANA TION)) (OPTIONAL  ) | It includes hiding the implementation part and showing only the required data and features to the user. It is done to hide the implementation complexity and details from the user. And to provide a good interface in programming. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Abstraction can apply to \_\_\_\_\_\_\_\_\_\_\_\_ |
| ((OPTION\_ A)) | Control and data |
| ((OPTION\_ B)) | Only data |
| ((OPTION\_ C)) | Only control |
| ((OPTION\_ D)) | Classes |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | a |
| ((EXPLANA TION)) (OPTIONAL  ) | Abstraction applies to both. Control abstraction involves use of subroutines and control flow abstraction. Data abstraction involves handling pieces of data in meaningful ways. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | When there is multiple IF and a single ELSE then the ELSE part doesn't get a clear view to go with which IF, this problem is called |
| ((OPTION\_ A)) | dangling if problem |
| ((OPTION\_ B)) | Multiple if problem |
| ((OPTION\_ C)) | dangling else problem |
| ((OPTION\_ D)) | None of the above |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | c |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | How is iteration controlled? |
| ((OPTION\_ A)) | Logical |
| ((OPTION\_ B)) | counter |
| ((OPTION\_ C)) | Both a and b |
| ((OPTION\_ D)) | None of the above |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | c |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | In Logically-Controlled Loops repetition control is based on |
| ((OPTION\_ A)) | counter |
| ((OPTION\_ B)) | Boolen experssion |
| ((OPTION\_ C)) | combination of the A & B |
| ((OPTION\_ D)) | None of the above |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | b |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Relational operator always have lower precedence than the arithmetic operators |
| ((OPTION\_ A)) | true |
| ((OPTION\_ B)) | false |
| ((OPTION\_ C)) |  |
| ((OPTION\_ D)) |  |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | a |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | In which type subscript ranges are statically bound, but the allocation is done at declaration elaboration time during execution |
| ((OPTION\_ A)) | Fixed stack-dynamic array |
| ((OPTION\_ B)) | Stack-dynamic array |
| ((OPTION\_ C)) | Fixed heap dynamic array |
| ((OPTION\_ D)) | Heap-dynamic array |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | a |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Binding of subscript ranges and storage allocation is dynamic and can change any number of times |
| ((OPTION\_ A)) | Fixed heap dynamic array |
| ((OPTION\_ B)) | Heap-dynamic array |
| ((OPTION\_ C)) | Stack-dynamic array |
| ((OPTION\_ D)) | Static array |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | b |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | An unordered collection of data elements that are indexed by an equal number of values called keys are called --------- |
| ((OPTION\_ A)) | rectangular array |
| ((OPTION\_ B)) | Jagged Arrays |
| ((OPTION\_ C)) | record |
| ((OPTION\_ D)) | Associative Arrays |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | d |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Heterogeneous Arrays supported by which language |
| ((OPTION\_ A)) | C |
| ((OPTION\_ B)) | Ada |
| ((OPTION\_ C)) | Python |
| ((OPTION\_ D)) | ML |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | c |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | C:\Users\Bhuri\Desktop\q1.png |
| ((OPTION\_ A)) | 1 |
| ((OPTION\_ B)) | 2 |
| ((OPTION\_ C)) | 3 |
| ((OPTION\_ D)) | 9 |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | D |
| ((EXPLANA TION)) (OPTIONAL  ) |  |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is the correct order of evaluation for the below expression? *z = x + y \* z / 4 % 2 - 1* |
| ((OPTION\_ A)) | \* / % + - = |
| ((OPTION\_ B)) | = \* / % + - |
| ((OPTION\_ C)) | / \* % - + = |
| ((OPTION\_ D)) | \* % / - + = |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | A |
| ((EXPLANA TION)) (OPTIONAL  ) | C uses left associativity for evaluating expressions to break a tie between two operators having same precedence |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | |  |  | | --- | --- | |  | Which of the following is the correct usage of conditional operators used in C? | |  | |
| ((OPTION\_ A)) | a>b ? c=30 : c=40; |
| ((OPTION\_ B)) | a>b ? c=30; |
| ((OPTION\_ C)) | max = a>b ? a>c?a:c:b>c?b:c |
| ((OPTION\_ D)) | return (a>b)?(a:b) |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) | Option A: assignment statements are always return in paranthesis in the case of conditional operator. It should be a>b? (c=30):(c=40);  Option B: it is syntatically wrong.  Option D: syntatically wrong, it should be return(a>b ? a:b);  Option C: it uses nested conditional operator, this is logic for finding greatest number out of three numbers. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) | Which of the following is the correct order if calling functions in the below code? *a = f1(23, 14) \* f2(12/4) + f3();* |
| ((OPTION\_ A)) | f1, f2, f3 |
| ((OPTION\_ B)) | f3, f2, f1 |
| ((OPTION\_ C)) | Order may vary from compiler to compiler |
| ((OPTION\_ D)) | None of above |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | C |
| ((EXPLANA TION)) (OPTIONAL  ) | Here, Multiplication will happen before the addition, but in which order the functions would be called is undefined. In an arithmetic expression the parenthesis tell the compiler which operands go with which operators but do not force the compiler to evaluate everything within the parenthesis first. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) |  |
| ((OPTION\_ A)) | 1,2 |
| ((OPTION\_ B)) | 1,3 |
| ((OPTION\_ C)) | 2,4 |
| ((OPTION\_ D)) | 1,2,3 |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | D |
| ((EXPLANA TION)) (OPTIONAL  ) | An operation with only one operand is called unary operation. **Unary operators**: *!* Logical NOT operator. *~* bitwise NOT operator. *sizeof* Size-of operator.  *&&* Logical AND is a logical operator.  Therefore, 1, 2, 3 are unary operators. |

|  |  |
| --- | --- |
| ((MARKS))  (1/2/3...) | 1 |
| ((QUESTIO N)) |  |
| ((OPTION\_ A)) | 2134 |
| ((OPTION\_ B)) | 1234 |
| ((OPTION\_ C)) | 4321 |
| ((OPTION\_ D)) | 3214 |
| ((CORRECT  \_CHOICE))  (A/B/C/D) | A |
| ((EXPLANA TION)) (OPTIONAL  ) | 2. Arithmetic operators: *\*, /, %, +, -*  1. Relational operators: *>, <, >=, <=, ==, !=* 3. Logical operators : *!, &&, ||* 4. Assignment operators: *=* |