1. How many times "IndiaBIX" is get printed?

```
#include<stdio.h>
int main()
{
    int x;
    for(x=-1; x<=10; x++)
    {
        if(x < 5)
            continue;
        else
            break;
        printf("IndiaBIX");
    }
    return 0;
}
```

A. Infinite times

B. 11 times

C. 0 times

D. 10 times

Answer: Option **C**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

2. How many times the `while` loop will get executed if a `short int` is 2 byte wide?

```
#include<stdio.h>
int main()
{
    int j=1;
    while(j <= 255)
    {
        printf("%c %d\n", j, j);
        j++;
    }
    return 0;
}
```

A. Infinite times

B. 255 times

C. 256 times

D. 254 times

Answer: Option **B**
Explanation:
The `while(j <= 255)` loop will get executed 255 times. The size short int(2 byte wide) does not affect the `while()` loop.

View Answer Discuss in Forum Workspace Report

---

3. Which of the following is not logical operator?

   A. &

   B. &&

   C. ||

   D. !

   Answer: Option **A**
   Explanation:
   Bitwise operators:
   `&` is a Bitwise AND operator.
   Logical operators:
   `&&` is a Logical AND operator.
   `||` is a Logical OR operator.
   `!` is a NOT operator.
   So, '`&`' is not a Logical operator.
   View Answer Discuss in Forum Workspace Report

---

4. In mathematics and computer programming, which is the correct order of mathematical operators ?

   A. Addition, Subtraction, Multiplication, Division

   B. Division, Multiplication, Addition, Subtraction

   C. Multiplication, Addition, Division, Subtraction

   D. Addition, Division, Modulus, Subtraction

   Answer: Option **B**
   Explanation:

   Simply called as BODMAS (Brackets, Order, Division, Multiplication, Addition and Subtraction).

   Mnemonics are often used to help students remember the rules, but the rules taught by the use of acronyms can be misleading. In the United States the acronym PEMDAS is common. It stands for Parentheses, Exponents, Multiplication, Division, Addition, Subtraction. In other English speaking countries, Parentheses may be called Brackets, or symbols of inclusion and Exponentiation may be called either Indices, Powers or Orders, and since multiplication and division are of equal precedence, M and D are often interchanged, leading to such acronyms as BEDMAS, BIDMAS, BODMAS, BERDMAS, PERDMAS, and BPODMAS.

   For more info: **http://en.wikipedia.org/wiki/Order_of_operations**
   View Answer Discuss in Forum Workspace Report

---

5. Which of the following cannot be checked in a `switch-case` statement?

   A. Character

B. Integer

C. Float

D. enum

Answer: Option **C**
Explanation:
The `switch/case` statement in the c language is defined by the language specification to use an `int` value, so you can not use a `float` value.

```
switch( expression )
{
    case constant-expression1:    statements 1;
    case constant-expression2:    statements 2;
    case constant-expression3:    statements3 ;
    ...
    ...
    default : statements 4;
}
```

The value of the '`expression`' in a switch-case statement must be an integer, char, short, long. Float and double are not allowed.

1. What will be the output of the program?

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

A. 0, 1, 2, 3, 4, 5

B. 5

C. 1, 2, 3, 4

D. 6

Answer: Option **D**
Explanation:
Step 1: `int i = 0;` here variable `i` is an integer type and initialized to '0'.
Step 2: `for(; i<=5; i++);` variable `i=0` is already assigned in previous step. The semi-colon at the end of this `for` loop tells, "there is no more statement is inside the loop".

Loop 1: here `i=0`, the condition in for(; 0<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 2: here `i=1`, the condition in for(; 1<=5; i++) loop satisfies and then `i` is incremented by

'1'(one)
Loop 3: here `i=2`, the condition in for(; 2<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 4: here `i=3`, the condition in for(; 3<=5; i++) loop satisfies and then `i` is increemented by '1'(one)
Loop 5: here `i=4`, the condition in for(; 4<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 6: here `i=5`, the condition in for(; 5<=5; i++) loop satisfies and then `i` is incremented by '1'(one)
Loop 7: here `i=6`, the condition in for(; 6<=5; i++) loop fails and then `i` is not incremented.
Step 3: `printf("%d", i);` here the value of `i` is 6. Hence the output is '6'.
View Answer Discuss in Forum Workspace Report

---

2. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    char str[]="C-program";
    int a = 5;
    printf(a >10?"Ps\n":"%s\n", str);
    return 0;
}
```

A. C-program

B. Ps

C. Error

D. None of above

Answer: Option **A**
Explanation:
Step 1: `char str[]="C-program";` here variable `str` contains "C-program".
Step 2: `int a = 5;` here variable `a` contains "5".
Step 3: `printf(a >10?"Ps\n":"%s\n", str);` this statement can be written as

```
if(a > 10)
{
    printf("Ps\n");
}
else
{
    printf("%s\n", str);
}
```

Here we are checking `a > 10` means `5 > 10`. Hence this condition will be failed. So it prints variable `str`.

Hence the output is "C-program".

View Answer Discuss in Forum Workspace Report

3. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int a = 500, b = 100, c;
    if(!a >= 400)
        b = 300;
    c = 200;
    printf("b = %d c = %d\n", b, c);
    return 0;
}
```

A. b = 300 c = 200

B. b = 100 c = garbage

C. b = 300 c = garbage

D. b = 100 c = 200

Answer: Option **D**
Explanation:
Initially variables `a = 500`, `b = 100` and `c` is not assigned.
Step 1: `if(!a >= 400)`
Step 2: `if(!500 >= 400)`
Step 3: `if(0 >= 400)`
Step 4: `if(FALSE)` Hence the `if` condition is failed.
Step 5: So, variable `c` is assigned to a value '200'.
Step 6: `printf("b = %d c = %d\n", b, c);` It prints value of `b` and `c`.
Hence the output is "b = 100 c = 200"
View Answer Discuss in Forum Workspace Report

4. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    unsigned int i = 65535; /* Assume 2 byte integer*/
    while(i++ != 0)
        printf("%d",++i);
    printf("\n");
    return 0;
}
```

A. Infinite loop

B. 0 1 2 ... 65535

C. 0 1 2 ... 32767 - 32766 -32765 -1 0

D. No output

Answer: Option **A**
Explanation:
Here `unsigned int` size is 2 bytes. It varies from 0,1,2,3, ... to 65535.
Step 1:`unsigned int i = 65535;`
Step 2:
Loop 1: `while(i++ != 0)` this statement becomes `while(65535 != 0)`. Hence the `while(TRUE)` condition is satisfied. Then the `printf("%d", ++i);` prints '1'(variable `'i'` is already incremented by '1' in while statement and now incremented by '1' in printf statement) **Loop 2**: `while(i++ != 0)` this statement becomes `while(1 != 0)`. Hence the `while(TRUE)` condition is satisfied. Then the `printf("%d", ++i);` prints '3'(variable `'i'` is already incremented by '1' in while statement and now incremented by '1' in printf statement)
....
....
The while loop will never stops executing, because variable `i` will never become '0'(zero). Hence it is an 'Infinite loop'.
View Answer Discuss in Forum Workspace Report

---

5. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x = 3;
    float y = 3.0;
    if(x == y)
        printf("x and y are equal");
    else
        printf("x and y are not equal");
    return 0;
}
```

A. x and y are equal

B. x and y are not equal

C. Unpredictable

D. No output

Answer: Option **A**
Explanation:
Step 1: `int x = 3;` here variable `x` is an integer type and initialized to '3'.
Step 2: `float y = 3.0;` here variable `y` is an float type and initialized to '3.0'
Step 3: `if(x == y)` here we are comparing `if(3 == 3.0)` hence this condition is satisfied. Hence it prints "x and y are equal".

6. What will be the output of the program, if a `short int` is 2 bytes wide?

```
#include<stdio.h>
int main()
{
    short int i = 0;
    for(i<=5 && i>=-1; ++i; i>0)
        printf("%u,", i);
    return 0;
}
```

A. 1 ... 65535

B. Expression syntax error

C. No output

D. 0, 1, 2, 3, 4, 5

Answer: Option **A**
Explanation:
`for(i<=5 && i>=-1; ++i; i>0)` so expression `i<=5 && i>=-1` initializes `for` loop.
expression `++i` is the loop condition. expression `i>0` is the increment expression.
In `for( i <= 5 && i >= -1; ++i; i>0)` expression `i<=5 && i>=-1` evaluates to one.
Loop condition always get evaluated to `true`. Also at this point it increases `i` by one.
An increment_expression `i>0` has no effect on value of `i`.so for loop get executed till the limit of integer (ie. 65535)
View Answer Discuss in Forum Workspace Report

---

7. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    char ch;
    if(ch = printf(""))
        printf("It matters\n");
    else
        printf("It doesn't matters\n");
    return 0;
}
```

A. It matters

B. It doesn't matters

C. matters

D. No output

Answer: Option **B**
Explanation:
`printf()` returns the number of charecters printed on the console.

Step 1: `if(ch = printf(""))` here `printf()` does not print anything, so it returns '0'(zero).
Step 2: `if(ch = 0)` here variable ch has the value '0'(zero).
Step 3: `if(0)` Hence the `if` condition is not satisfied. So it prints the `else` statements.
Hence the output is "It doesn't matters".

Note: Compiler shows a warning "possibly incorrect assinment".

View Answer Discuss in Forum Workspace Report

---

8. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    unsigned int i = 65536; /* Assume 2 byte integer*/
    while(i != 0)
        printf("%d",++i);
    printf("\n");
    return 0;
}
```

A. Infinite loop

B. 0 1 2 ... 65535

C. 0 1 2 ... 32767 - 32766 -32765 -1 0

D. No output

Answer: Option **D**
Explanation:
Here `unsigned int` size is 2 bytes. It varies from 0,1,2,3, ... to 65535.
Step 1:`unsigned int i = 65536;` here variable `i` becomes '0'(zero). because `unsigned int` varies from 0 to 65535.
Step 2: `while(i != 0)` this statement becomes `while(0 != 0)`. Hence the `while(FALSE)` condition is not satisfied. So, the inside the statements of `while` loop will not get executed.

Hence there is no output.

Note: Don't forget that the size of int should be 2 bytes. If you run the above program in GCC it may run infinite loop, because in Linux platform the size of the integer is 4 bytes.

View Answer Discuss in Forum Workspace Report

---

9. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    float a = 0.7;
    if(0.7 > a)
        printf("Hi\n");
    else
```

```
        printf("Hello\n");
    return 0;
}
```

A. Hi

B. Hello

C. Hi Hello

D. None of above

Answer: Option **A**
Explanation:
`if(0.7 > a)` here `a` is a float variable and `0.7` is a double constant. The double constant `0.7` is greater than the float variable `a`. Hence the `if` condition is satisfied and it prints `'Hi'`
Example:

```
#include<stdio.h>
int main()
{
    float a=0.7;
    printf("%.10f %.10f\n",0.7, a);
    return 0;
}
```

Output:
0.7000000000 0.6999999881
View Answer Discuss in Forum Workspace Report

10. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int a=0, b=1, c=3;
    *((a) ? &b : &a) = a ? b : c;
    printf("%d, %d, %d\n", a, b, c);
    return 0;
}
```

A. 0, 1, 3

B. 1, 2, 3

C. 3, 1, 3

D. 1, 3, 1

Answer: Option **C**
Explanation:
Step 1: `int a=0, b=1, c=3;` here variable `a`, `b`, and `c` are declared as integer type and initialized to 0, 1, 3 respectively.
Step 2: `*((a) ? &b : &a) = a ? b : c;` The right side of the expression `(a?b:c)` becomes `(0?1:3)`. Hence it return the value '3'.

The left side of the expression `*((a) ? &b : &a)` becomes `*((0) ? &b : &a)`. Hence this contains the address of the variable a `*(&a)`.
Step 3: `*((a) ? &b : &a) = a ? b : c;` Finally this statement becomes *(&a)=3. Hence the variable `a` has the value '3'.
Step 4: `printf("%d, %d, %d\n", a, b, c);` It prints "3, 1, 3".

11. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int k, num = 30;
    k = (num < 10) ? 100 : 200;
    printf("%d\n", num);
    return 0;
}
```

A. 200

B. 30

C. 100

D. 500

Answer: Option **B**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

12. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int a = 300, b, c;
    if(a >= 400)
        b = 300;
    c = 200;
    printf("%d, %d, %d\n", a, b, c);
    return 0;
}
```

A. 300, 300, 200

B. Garbage, 300, 200

C. 300, Garbage, 200

D. 300, 300, Garbage

Answer: Option **C**
Explanation:

Step 1: `int a = 300, b, c;` here variable `a` is initialized to '300', variable b and c are declared, but not initialized.
Step 2: `if(a >= 400)` means `if(300 >= 400)`. Hence this condition will be failed.
Step 3: `c = 200;` here variable `c` is initialized to '200'.
Step 4: `printf("%d, %d, %d\n", a, b, c);` It prints "300, garbage value, 200". because variable `b` is not initialized.
View Answer Discuss in Forum Workspace Report

---

13. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x=1, y=1;
    for(; y; printf("%d %d\n", x, y))
    {
        y = x++ <= 5;
    }
    printf("\n");
    return 0;
}
```

A.
2 1
3 1
4 1
5 1
6 1
7 0

B.
2 1
3 1
4 1
5 1
6 1

C.
2 1
3 1
4 1
5 1

D.
2 2
3 3
4 4
5 5

Answer: Option **A**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

14. What will be the output of the program?

```c
#include<stdio.h>
int main()
{
    int i = 5;
    while(i-- >= 0)
        printf("%d,", i);
    i = 5;
    printf("\n");
    while(i-- >= 0)
        printf("%i,", i);
    while(i-- >= 0)
        printf("%d,", i);
    return 0;
}
```

A. 4, 3, 2, 1, 0, -1
4, 3, 2, 1, 0, -1

B. 5, 4, 3, 2, 1, 0
5, 4, 3, 2, 1, 0

C. Error

D. 5, 4, 3, 2, 1, 0
5, 4, 3, 2, 1, 0
5, 4, 3, 2, 1, 0

Answer: Option **A**
Explanation:
Step 1: Initially the value of variable `i` is '5'.
Loop 1: `while(i-- >= 0)` here i = 5, this statement becomes `while(5-- >= 0)` Hence the `while` condition is satisfied and it prints '4'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 2: `while(i-- >= 0)` here i = 4, this statement becomes `while(4-- >= 0)` Hence the `while` condition is satisfied and it prints '3'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 3: `while(i-- >= 0)` here i = 3, this statement becomes `while(3-- >= 0)` Hence the `while` condition is satisfied and it prints '2'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 4: `while(i-- >= 0)` here i = 2, this statement becomes `while(2-- >= 0)` Hence the `while` condition is satisfied and it prints '1'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 5: `while(i-- >= 0)` here i = 1, this statement becomes `while(1-- >= 0)` Hence the `while` condition is satisfied and it prints '0'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 6: `while(i-- >= 0)` here i = 0, this statement becomes `while(0-- >= 0)` Hence the `while` condition is satisfied and it prints '-1'. (variable `'i'` is decremented by `'1'`(one) in previous while condition)
Loop 7: `while(i-- >= 0)` here i = -1, this statement becomes `while(-1-- >= 0)` Hence the `while` condition is not satisfied and loop exits.
The output of first while loop is 4,3,2,1,0,-1

Step 2: Then the value of variable `i` is initialized to '5' Then it prints a new line character(\n). See the above Loop 1 to Loop 7 .
The output of second while loop is 4,3,2,1,0,-1
Step 3: The third while loop, `while(i-- >= 0)` here i = -1(because the variable `'i'` is decremented to '-1' by previous while loop and it never initialized.). This statement becomes `while(-1-- >= 0)` Hence the `while` condition is not satisfied and loop exits.
Hence the output of the program is
4,3,2,1,0,-1
4,3,2,1,0,-1
View Answer Discuss in Forum Workspace Report

---

15. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=3;
    switch(i)
    {
        case 1:
            printf("Hello\n");
        case 2:
            printf("Hi\n");
        case 3:
            continue;
        default:
            printf("Bye\n");
    }
    return 0;
}
```

A. Error: Misplaced continue

B. Bye

C. No output

D. Hello Hi

Answer: Option **A**
Explanation:
The keyword `continue` cannot be used in `switch case`. It must be used in `for` or `while` or `do while` loop. If there is any looping statement in switch case then we can use `continue`.

16. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x = 10, y = 20;
    if(!(!x) && x)
        printf("x = %d\n", x);
    else
        printf("y = %d\n", y);
    return 0;
}
```

A. y =20

B. x = 0

C. x = 10

D. x = 1

Answer: Option **C**

Explanation:

The `logical not` operator takes expression and evaluates to true if the expression is false and evaluates to false if the expression is true. In other words it reverses the value of the expression.

Step 1: `if(!(!x) && x)`
Step 2: `if(!(!10) && 10)`
Step 3: `if(!(0) && 10)`
Step 3: `if(1 && 10)`
Step 4: `if(TRUE)` here the `if` condition is satisfied. Hence it prints `x = 10`.

View Answer Discuss in Forum Workspace Report

---

17. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=4;
    switch(i)
    {
        default:
           printf("This is default\n");
        case 1:
           printf("This is case 1\n");
           break;
        case 2:
           printf("This is case 2\n");
           break;
        case 3:
           printf("This is case 3\n");
    }
    return 0;
}
```

A. This is default
This is case 1

B. This is case 3
This is default

C. This is case 1
This is case 3

D. This is default

Answer: Option **A**
Explanation:
In the very begining of switch-case statement `default` statement is encountered. So, it prints "This is default".
In `default` statement there is no `break;` statement is included. So it prints the `case 1` statements. "This is case 1".
Then the `break;` statement is encountered. Hence the program exits from the switch-case block.
View Answer Discuss in Forum Workspace Report

---

18. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i = 1;
    switch(i)
    {
        printf("Hello\n");
        case 1:
            printf("Hi\n");
            break;
        case 2:
            printf("\nBye\n");
            break;
    }
    return 0;
}
```

A. Hello
Hi

B. Hello
Bye

C. Hi

D. Bye

Answer: Option **C**
Explanation:
`switch(i)` has the variable `i` it has the value '1'(one).
Then `case 1:` statements got executed. so, it prints "Hi". The `break;` statement make the program to be exited from switch-case statement.

`switch-case` do not execute any statements outside these blocks `case` and `default`

Hence the output is "Hi".

View Answer Discuss in Forum Workspace Report

---

19. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    char j=1;
    while(j < 5)
    {
        printf("%d, ", j);
        j = j+1;
    }
    printf("\n");
    return 0;
}
```

A. 1 2 3 ... 127

B. 1 2 3 ... 255

C. 1 2 3 ... 127 128 0 1 2 3 ... infinite times

D. 1, 2, 3, 4

Answer: Option **D**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

20. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x, y, z;
    x=y=z=1;
    z = ++x || ++y && ++z;
    printf("x=%d, y=%d, z=%d\n", x, y, z);
    return 0;
}
```

A. x=2, y=1, z=1

B. x=2, y=2, z=1

C. x=2, y=2, z=2

D. x=1, y=2, z=1

Answer: Option **A**

Explanation:
Step 1: `x=y=z=1;` here the variables `x` ,`y`, `z` are initialized to value '1'.
Step 2: `z = ++x || ++y && ++z;` becomes `z = ( (++x) || (++y && ++z) )`.
Here `++x` becomes 2. So there is no need to check the other side because ||(Logical OR) condition is satisfied.(z = (2 || ++y && ++z)). There is no need to process ++y && ++z. Hence it returns '1'.
So the value of variable `z` is '1'
Step 3: `printf("x=%d, y=%d, z=%d\n", x, y, z);` It prints "x=2, y=1, z=1". here `x` is increemented in previous step. `y` and `z` are not increemented.

1. Point out the error, if any in the `for` loop.

```
#include<stdio.h>
int main()
{
    int i=1;
    for(;;)
    {
        printf("%d\n", i++);
        if(i>10)
           break;
    }
    return 0;
}
```

A. There should be a condition in the `for` loop

B. The two semicolons should be dropped

C. The `for` loop should be replaced with `while` loop.

D. No error

Answer: Option **D**
Explanation:
Step 1: `for(;;)` this statement will genereate infinite loop.
Step 2: `printf("%d\n", i++);` this statement will print the value of variable `i` and increement `i` by 1(one).
Step 3: `if(i>10)` here, if the variable `i` value is greater than 10, then the `for` loop breaks.
Hence the output of the program is
1
2
3
4
5
6
7
8
9
10
View Answer Discuss in Forum Workspace Report

2. Point out the error, if any in the program.

```
#include<stdio.h>
int main()
{
    int a = 10;
    switch(a)
    {
    }
    printf("This is c program.");
    return 0;
}
```

A. Error: No `case` statement specified

B. Error: No `default` specified

C. No Error

D. Error: infinite loop occurs

Answer: Option **C**
Explanation:
There can exists a `switch` statement, which has no `case`.
View Answer Discuss in Forum Workspace Report

---

3. Point out the error, if any in the program.

```
#include<stdio.h>
int main()
{
    int i = 1;
    switch(i)
    {
        printf("This is c program.");
        case 1:
            printf("Case1");
            break;
        case 2:
            printf("Case2");
            break;
    }
return 0;
}
```

A. Error: No default specified

B. Error: Invalid `printf` statement after `switch` statement

C. No Error and prints "Case1"

D. None of above

Answer: Option **C**
Explanation:

`switch(i)` becomes `switch(1)`, then the `case 1:` block is get executed. Hence it prints "Case1".
`printf("This is c program.");` is ignored by the compiler.

Hence there is no error and prints "Case1".

View Answer Discuss in Forum Workspace Report

---

4. Point out the error, if any in the `while` loop.

```
#include<stdio.h>
int main()
{
    int i=1;
    while()
    {
        printf("%d\n", i++);
        if(i>10)
           break;
    }
    return 0;
}
```

A. There should be a condition in the `while` loop

B. There should be at least a semicolon in the `while`

C. The `while` loop should be replaced with `for` loop.

D. No error

Answer: Option **A**
Explanation:
The `while()` loop must have conditional expression or it shows "Expression syntax" error.
Example: `while(i > 10){ ... }`
View Answer Discuss in Forum Workspace Report

---

5. Which of the following errors would be reported by the compiler on compiling the program given below?

```
#include<stdio.h>
int main()
{
    int a = 5;
    switch(a)
    {
    case 1:
    printf("First");

    case 2:
    printf("Second");

    case 3 + 2:
    printf("Third");
```

```
    case 5:
    printf("Final");
    break;

    }
    return 0;
}
```

A. There is no `break` statement in each case.

B. Expression as in `case 3 + 2` is not allowed.

C. Duplicate case `case 5:`

D. No error will be reported.

Answer: Option **C**
Explanation:
Because, `case 3 + 2:` and `case 5:` have the same constant value 5.

6. Point out the error, if any in the program.

```
#include<stdio.h>
int main()
{
    int P = 10;
    switch(P)
    {
       case 10:
       printf("Case 1");

       case 20:
       printf("Case 2");
       break;

       case P:
       printf("Case 2");
       break;
    }
    return 0;
}
```

A. Error: No default value is specified

B. Error: Constant expression required at line `case P:`

C. Error: There is no `break` statement in each case.

D. No error will be reported.

Answer: Option **B**
Explanation:

The compiler will report the error "Constant expression required" in the line `case P:` . Because, variable names cannot be used with `case` statements.
The `case` statements will accept only constant expression.
View Answer Discuss in Forum Workspace Report

---

7. Point out the error, if any in the program.

```
#include<stdio.h>
int main()
{
    int i = 1;
    switch(i)
    {
        case 1:
           printf("Case1");
           break;
        case 1*2+4:
           printf("Case2");
           break;
    }
return 0;
}
```

A. Error: in `case 1*2+4` statement

B. Error: No default specified

C. Error: in `switch` statement

D. No Error

Answer: Option **D**
Explanation:
Constant expression are accepted in `switch`

It prints "Case1"

View Answer Discuss in Forum Workspace Report

---

8. Point out the error, if any in the `while` loop.

```
#include<stdio.h>
int main()
{
    void fun();
    int i = 1;
    while(i <= 5)
    {
        printf("%d\n", i);
        if(i>2)
            goto here;
    }
return 0;
}
```

```
void fun()
{
    here:
    printf("It works");
}
```

A. No Error: prints "It works"

B. Error: `fun()` cannot be accessed

C. Error: `goto` cannot takeover control to other function

D. No error

Answer: Option **C**
Explanation:
A label is used as the target of a `goto` statement, and that label must be within the same function as the `goto` statement.
Syntax: goto <identifier> ;
Control is unconditionally transferred to the location of a local label specified by <identifier>.
Example:

```
#include <stdio.h>
int main()
{
    int i=1;
    while(i>0)
    {
        printf("%d", i++);
        if(i==5)
          goto mylabel;
    }
    mylabel:
    return 0;
}
```

Output: 1,2,3,4
View Answer Discuss in Forum Workspace Report

---

9. Point out the error, if any in the program.

```
#include<stdio.h>
int main()
{
    int a = 10, b;
    a >=5 ? b=100: b=200;
    printf("%d\n", b);
    return 0;
}
```

A. 100

B. 200

C. Error: L value required for b

D. Garbage value

Answer: Option **C**
Explanation:

Variable b is not assigned.

It should be like:

```
b = a >= 5 ? 100 : 200;
```

1. Which of the following statements are correct about the below program?

```
#include<stdio.h>
int main()
{
    int i = 10, j = 20;
    if(i = 5) && if(j = 10)
        printf("Have a nice day");
    return 0;
}
```

A. Output: Have a nice day

B. No output

C. Error: Expression syntax

D. Error: Undeclared identifier `if`

Answer: Option **C**
Explanation:
"Expression syntax" error occur in this line `if(i = 5) && if(j = 10)`.
It should be like `if((i == 5) && (j == 10))`.
View Answer Discuss in Forum Workspace Report

---

2. Which of the following statements are correct about the below program?

```
#include<stdio.h>
int main()
{
    int i = 10, j = 15;
    if(i % 2 = j % 3)
        printf("IndiaBIX\n");
    return 0;
}
```

A. Error: Expression syntax

B. Error: Lvalue required

C. Error: Rvalue required

D. The Code runs successfully

Answer: Option **B**
Explanation:
`if(i % 2 = j % 3)` This statement generates "LValue required error". There is no variable on the left side of the expression to assign `(j % 3)`.
View Answer Discuss in Forum Workspace Report

---

3. Which of the following statements are correct about the program?

```
#include<stdio.h>
int main()
{
    int x = 30, y = 40;
    if(x == y)
        printf("x is equal to y\n");

    else if(x > y)
        printf("x is greater than y\n");

    else if(x < y)
        printf("x is less than y\n")
    return 0;
}
```

A. Error: Statement missing

B. Error: Expression syntax

C. Error: Lvalue required

D. Error: Rvalue required

Answer: Option **A**
Explanation:

This program will result in error "Statement missing ;"

`printf("x is less than y\n")` here ; should be added to the end of this statement.
View Answer Discuss in Forum Workspace Report

---

4. Which of the following statements are correct about an `if-else` statements in a C-program?
   1: Every `if-else` statement can be replaced by an equivalent statements using `?:` operators
   2: Nested `if-else` statements are allowed.
   3: Multiple statements in an `if` block are allowed.
   4: Multiple statements in an `else` block are allowed.

A. 1 and 2

B. 2 and 3

C. 1, 2 and 4

D. 2, 3, 4

Answer: Option **D**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

5. Which of the following statements are correct about the below program?

```
#include<stdio.h>
int main()
{
    int i = 0;
    i++;
    if(i <= 5)
    {
        printf("IndiaBIX\n");
        exit(0);
        main();
    }
    return 0;
}
```

A. The program prints 'IndiaBIX' 5 times

B. The program prints 'IndiaBIX' one time

C. The call to `main()` after `exit()` doesn't materialize.

D. The compiler reports an error since `main()` cannot call itself.

Answer: Option **B**
Explanation:
Step 1: `int i = 0;` here variable `i` is declared as an integer type and initialized to '0'(zero).
Step 2: `i++;` here variable `i` is increemented by 1(one). Hence, i = 1
Step 3: `if(i <= 5)` becomes `if(1 <= 5)` here we are checking '1' is less than or equal to '5'. Hence the if condition is satisfied.
Step 4: `printf("IndiaBIX\n");` It prints "IndiaBIX"
Step 5: `exit();` terminates the program execution.

Hence the output is "IndiaBIX".

6. Which of the following statements are correct about the below C-program?

```
#include<stdio.h>
int main()
{
    int x = 10, y = 100%90, i;
    for(i=1; i<10; i++)
    if(x != y);
```

```
        printf("x = %d y = %d\n", x, y);
    return 0;
}
```

1 : The `printf()` function is called 10 times.

2 : The program will produce the output `x = 10 y = 10`

3 : The `;` after the `if(x!=y)` will NOT produce an error.

4 : The program will not produce output.

A. 1

B. 2, 3

C. 3, 4

D. 4

Answer: Option **B**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

7. Which of the following sentences are correct about a `for` loop in a C program?

1: `for` loop works faster than a `while` loop.

2: All things that can be done using a `for` loop can also be done using a `while` loop.

3: `for(;;);` implements an infinite loop.

4: `for` loop can be used if we want statements in a loop get executed at least once.

A. 1

B. 1, 2

C. 2, 3

D. 2, 3, 4

Answer: Option **D**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

8. Which of the following statements are correct about the below program?

```
#include<stdio.h>
int main()
{
    int n = 0, y = 1;
    y == 1 ? n=0 : n=1;
    if(n)
        printf("Yes\n");
```

```
    else
        printf("No\n");
    return 0;
}
```

A. Error: Declaration terminated incorrectly

B. Error: Syntax error

C. Error: Lvalue required

D. None of above

Answer: Option **C**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

9. Which of the following sentences are correct about a `switch` loop in a C program?

1: `switch` is useful when we wish to check the value of variable against a particular set of values.

2: `switch` is useful when we wish to check whether a value falls in different ranges.

3: Compiler implements a jump table for cases used in `switch`.

4: It is not necessary to use a `break` in every `switch` statement.

A. 1,2

B. 1,3,4

C. 2,4

D. 2

Answer: Option **B**