1. The keyword used to transfer control from a function back to the calling function is

   A. switch

   B. goto

   C. go back

   D. return

   Answer: Option **D**
   Explanation:
   The keyword `return` is used to transfer control from a function back to the calling function.
   Example:

```
#include<stdio.h>
int add(int, int); /* Function prototype */

int main()
{
    int a = 4, b = 3, c;
    c = add(a, b);
    printf("c = %d\n", c);
    return 0;
}
int add(int a, int b)
{
/* returns the value and control back to main() function */
   return (a+b);
}
```

   Output:
   c = 7
   View Answer Discuss in Forum Workspace Report

---

2. What is the notation for following functions?

```
1.  int f(int a, float b)
    {
        /* Some code */
    }

2.  int f(a, b)
    int a; float b;
    {
        /* Some code */
    }
```

   A. 1. KR Notation
   2. ANSI Notation

   B. 1. Pre ANSI C Notation
   2. KR Notation

C. 1. ANSI Notation
2. KR Notation

D. 1. ANSI Notation
2. Pre ANSI Notation

Answer: Option **C**
Explanation:
KR Notation means Kernighan and Ritche Notation.
View Answer Discuss in Forum Workspace Report

---

3. How many times the program will print "IndiaBIX" ?

```
#include<stdio.h>

int main()
{
    printf("IndiaBIX");
    main();
    return 0;
}
```

A. Infinite times

B. 32767 times

C. 65535 times

D. Till stack overflows

Answer: Option **D**
Explanation:
A `call stack` or `function stack` is used for several related purposes, but the main reason for having one is to keep track of the point to which each active subroutine should return control when it finishes executing.
A `stack overflow` occurs when too much memory is used on the call stack.
Here function `main()` is called repeatedly and its return address is stored in the stack. After stack memory is full. It shows stack overflow error.

1. What will be the output of the program in 16 bit platform (Turbo C under DOS)?

```
#include<stdio.h>

int main()
{
    int fun();
    int i;
    i = fun();
    printf("%d\n", i);
    return 0;
}
int fun()
{
```

```
    _AX = 1990;
}
```

A. Garbage value

B. 0 (Zero)

C. 1990

D. No output

Answer: Option **C**
Explanation:
Turbo C (Windows): The return value of the function is taken from the Accumulator `_AX=1990`.

But it may not work as expected in GCC compiler (Linux).

View Answer Discuss in Forum Workspace Report

---

2. What will be the output of the program?

```
#include<stdio.h>
void fun(int*, int*);
int main()
{
    int i=5, j=2;
    fun(&i, &j);
    printf("%d, %d", i, j);
    return 0;
}
void fun(int *i, int *j)
{
    *i = *i**i;
    *j = *j**j;
}
```

A. 5, 2

B. 10, 4

C. 2, 5

D. 25, 4

Answer: Option **D**
Explanation:
Step 1: `int i=5, j=2;` Here variable `i` and `j` are declared as an integer type and initialized to 5 and 2 respectively.
Step 2: `fun(&i, &j);` Here the function `fun()` is called with two parameters `&i` and `&j` (The & denotes `call by reference`. So the address of the variable `i` and `j` are passed.)
Step 3: `void fun(int *i, int *j)` This function is called by reference, so we have to use `*` before the parameters.
Step 4: `*i = *i**i;` Here `*i` denotes the value of the variable `i`. We are multiplying `5*5` and storing the result `25` in same variable `i`.
Step 5: `*j = *j**j;` Here `*j` denotes the value of the variable `j`. We are multiplying `2*2` and storing the result `4` in same variable `j`.

Step 6: Then the function `void fun(int *i, int *j)` return back the control back to `main()` function.
Step 7: `printf("%d, %d", i, j);` It prints the value of variable `i` and `j`.

Hence the output is 25, 4.

View Answer Discuss in Forum Workspace Report

---

3. What will be the output of the program?

```
#include<stdio.h>
int i;
int fun();

int main()
{
    while(i)
    {
        fun();
        main();
    }
    printf("Hello\n");
    return 0;
}
int fun()
{
    printf("Hi");
}
```

A. Hello

B. Hi Hello

C. No output

D. Infinite loop

Answer: Option **A**
Explanation:
Step 1: `int i;` The variable `i` is declared as an integer type.
Step 1: `int fun();` This prototype tells the compiler that the function `fun()` does not accept any arguments and it returns an integer value.
Step 1: `while(i)` The value of `i` is not initialized so this while condition is failed. So, it does not execute the `while` block.
Step 1: `printf("Hello\n");` It prints "Hello".

Hence the output of the program is "Hello".

View Answer Discuss in Forum Workspace Report

---

4. What will be the output of the program?

```
#include<stdio.h>
int reverse(int);
```

```
int main()
{
    int no=5;
    reverse(no);
    return 0;
}
int reverse(int no)
{
    if(no == 0)
        return 0;
    else
        printf("%d,", no);
    reverse (no--);
}
```

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

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

C. Print 5, 4, 3, 2, 1, 0

D. Infinite loop

Answer: Option **D**

Explanation:

Step 1: `int no=5;` The variable `no` is declared as integer type and initialized to 5.
Step 2: `reverse(no);` becomes `reverse(5);` It calls the function `reverse()` with '5' as parameter.
The function reverse accept an integer number 5 and it returns '0'(zero) if(5 == 0) if the given number is '0'(zero) or else `printf("%d,", no);` it prints that number 5 and calls the function `reverse(5);`.
The function runs infinetely because the there is a post-decrement operator is used. It will not decrease the value of 'n' before calling the reverse() function. So, it calls `reverse(5)` infinitely.
Note: If we use pre-decrement operator like `reverse(--n)`, then the output will be 5, 4, 3, 2, 1. Because before calling the function, it decrements the value of 'n'.
View Answer Discuss in Forum Workspace Report

5. What will be the output of the program?

```
#include<stdio.h>
void fun(int);
typedef int (*pf) (int, int);
int proc(pf, int, int);

int main()
{
    int a=3;
    fun(a);
    return 0;
}
void fun(int n)
{
    if(n > 0)
    {
        fun(--n);
        printf("%d,", n);
```

```
        fun(--n);
    }
}
```

A. 0, 2, 1, 0,

B. 1, 1, 2, 0,

C. 0, 1, 0, 2,

D. 0, 1, 2, 0,

Answer: Option **D**

6. What will be the output of the program?

```
#include<stdio.h>
int sumdig(int);
int main()
{
    int a, b;
    a = sumdig(123);
    b = sumdig(123);
    printf("%d, %d\n", a, b);
    return 0;
}
int sumdig(int n)
{
    int s, d;
    if(n!=0)
    {
        d = n%10;
        n = n/10;
        s = d+sumdig(n);
    }
    else
        return 0;
    return s;
}
```

A. 4, 4

B. 3, 3

C. 6, 6

D. 12, 12

Answer: Option **C**

Explanation:

No answer description available for this question. Let us discuss.

View Answer Discuss in Forum Workspace Report

7. What will be the output of the program?

```
#include<stdio.h>

int main()
{
    void fun(char*);
    char a[100];
    a[0] = 'A'; a[1] = 'B';
    a[2] = 'C'; a[3] = 'D';
    fun(&a[0]);
    return 0;
}
void fun(char *a)
{
    a++;
    printf("%c", *a);
    a++;
    printf("%c", *a);
}
```

A. AB

B. BC

C. CD

D. No output

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

8. What will be the output of the program?

```
#include<stdio.h>

int main()
{
    int fun(int);
    int i = fun(10);
    printf("%d\n", --i);
    return 0;
}
int fun(int i)
{
   return (i++);
}
```

A. 9

B. 10

C. 11

D. 8

Answer: Option **A**
Explanation:
Step 1: `int fun(int);` Here we declare the prototype of the function `fun()`.
Step 2: `int i = fun(10);` The variable `i` is declared as an integer type and the result of the `fun(10)` will be stored in the variable `i`.
Step 3: `int fun(int i){ return (i++); }` Inside the `fun()` we are returning a value return(i++). It returns 10. because `i++` is the post-increement operator.
Step 4: Then the control back to the `main` function and the value 10 is assigned to variable `i`.
Step 5: `printf("%d\n", --i);` Here `--i` denoted pre-increement. Hence it prints the value 9.
View Answer Discuss in Forum Workspace Report

---

9. What will be the output of the program?

```
#include<stdio.h>
int check (int, int);

int main()
{
    int c;
    c = check(10, 20);
    printf("c=%d\n", c);
    return 0;
}
int check(int i, int j)
{
    int *p, *q;
    p=&i;
    q=&j;
    i>=45 ? return(*p): return(*q);
}
```

A. Print 10

B. Print 20

C. Print 1

D. Compile error

Answer: Option **D**
Explanation:
There is an error in this line `i>=45 ? return(*p): return(*q);`. We cannot use `return` keyword in the terenary operators.
View Answer Discuss in Forum Workspace Report

---

10. What will be the output of the program?

```
#include<stdio.h>
int fun(int, int);
```

```
typedef int (*pf) (int, int);
int proc(pf, int, int);

int main()
{
    printf("%d\n", proc(fun, 6, 6));
    return 0;
}
int fun(int a, int b)
{
   return (a==b);
}
int proc(pf p, int a, int b)
{
   return ((*p)(a, b));
}
```

A. 6

B. 1

C. 0

D. -1

Answer: Option **B**

11. What will be the output of the program?

```
#include<stdio.h>

int main()
{
    int i=1;
    if(!i)
        printf("IndiaBIX,");
    else
    {
        i=0;
        printf("C-Program");
        main();
    }
    return 0;
}
```

A. prints "IndiaBIX, C-Program" infinitely

B. prints "C-Program" infinetly

C. prints "C-Program, IndiaBIX" infinitely

D. Error: `main()` should not inside `else` statement

Answer: Option **B**
Explanation:
Step 1: `int i=1;` The variable `i` is declared as an integer type and initialized to 1(one).
Step 2: `if(!i)` Here the `!`(NOT) operator reverts the `i` value 1 to 0. Hence the `if(0)` condition fails. So it goes to else part.
Step 3: `else { i=0;` In the else part variable `i` is assigned to value 0(zero).
Step 4: `printf("C-Program");` It prints the "C-program".
Step 5: `main();` Here we are calling the `main()` function.
After calling the function, the program repeats from **step 1** to **step 5** infinitely.

Hence it prints "C-Program" infinitely.

View Answer Discuss in Forum Workspace Report

---

12. What will be the output of the program?

```
#include<stdio.h>

int addmult(int ii, int jj)
{
    int kk, ll;
    kk = ii + jj;
    ll = ii * jj;
    return (kk, ll);
}

int main()
{
    int i=3, j=4, k, l;
    k = addmult(i, j);
    l = addmult(i, j);
    printf("%d %d\n", k, l);
    return 0;
}
```

A. 12 12

B. No error, No output

C. Error: Compile error

D. None of above

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

---

13. What will be the output of the program?

```
#include<stdio.h>
int i;
int fun1(int);
```

```
int fun2(int);

int main()
{
    extern int j;
    int i=3;
    fun1(i);
    printf("%d,", i);
    fun2(i);
    printf("%d", i);
    return 0;
}
int fun1(int j)
{
    printf("%d,", ++j);
    return 0;
}
int fun2(int i)
{
    printf("%d,", ++i);
    return 0;
}
int j=1;
```

A. 3, 4, 4, 3

B. 4, 3, 4, 3

C. 3, 3, 4, 4

D. 3, 4, 3, 4

Answer: Option **B**

Explanation:

Step 1: `int i;` The variable i is declared as an global and integer type.
Step 2: `int fun1(int);` This prototype tells the compiler that the `fun1()` accepts the one integer parameter and returns the integer value.
Step 3: `int fun2(int);` This prototype tells the compiler that the `fun2()` accepts the one integer parameter and returns the integer value.
Step 4: `extern int j;` Inside the main function, the `extern` variable `j` is declared and defined in another source file.
Step 5: `int i=3;` The local variable `i` is defines as an integer type and initialized to 3.
Step 6: `fun1(i);` The `fun1(i)` increements the given value of variable `i` prints it.
Here `fun1(i)` becomes `fun1(3)` hence it prints '4' then the control is given back to the `main` function.
Step 7: `printf("%d,", i);` It prints the value of local variable `i`. So, it prints '3'.
Step 8: `fun2(i);` The `fun2(i)` increements the given value of variable `i` prints it.
Here `fun2(i)` becomes `fun2(3)` hence it prints '4' then the control is given back to the `main` function.
Step 9: `printf("%d,", i);` It prints the value of local variable `i`. So, it prints '3'.

Hence the output is "4 3 4 3".

View Answer Discuss in Forum Workspace Report

14. What will be the output of the program?

```
#include<stdio.h>
int func1(int);

int main()
{
    int k=35;
    k = func1(k=func1(k=func1(k)));
    printf("k=%d\n", k);
    return 0;
}
int func1(int k)
{
    k++;
    return k;
}
```

A. k=35

B. k=36

C. k=37

D. k=38

Answer: Option **D**
Explanation:
Step 1: `int k=35;` The variable k is declared as an integer type and initialized to 35.
Step 2: `k = func1(k=func1(k=func1(k)));` The `func1(k)` increement the value of `k` by 1 and return it. Here the `func1(k)` is called 3 times. Hence it increements value of `k` = 35 to 38. The result is stored in the variable `k` = 38.
Step 3: `printf("k=%d\n", k);` It prints the value of variable `k` "38".
View Answer Discuss in Forum Workspace Report

15. What will be the output of the program?

```
#include<stdio.h>

int addmult(int ii, int jj)
{
    int kk, ll;
    kk = ii + jj;
    ll = ii * jj;
    return (kk, ll);
}

int main()
{
    int i=3, j=4, k, l;
    k = addmult(i, j);
    l = addmult(i, j);
    printf("%d, %d\n", k, l);
    return 0;
}
```

A. 12, 12

B. 7, 7

C. 7, 12

D. 12, 7

Answer: Option **A**
Explanation:
Step 1: `int i=3, j=4, k, l;` The variables `i, j, k, l` are declared as an integer type and variable `i`, `j` are initialized to 3, 4 respectively.

The function addmult(i, j); accept 2 integer parameters.

Step 2: `k = addmult(i, j);` becomes `k = addmult(3, 4)`
In the function `addmult()`. The variable `kk, ll` are declared as an integer type `int kk, ll;`
`kk = ii + jj;` becomes `kk = 3 + 4` Now the `kk` value is '7'.
`ll = ii * jj;` becomes `ll = 3 * 4` Now the `ll` value is '12'.
`return (kk, ll);` It returns the value of variable `ll` only.
The value 12 is stored in variable `'k'`.
Step 3: `l = addmult(i, j);` becomes `l = addmult(3, 4)`
`kk = ii + jj;` becomes `kk = 3 + 4` Now the `kk` value is '7'.
`ll = ii * jj;` becomes `ll = 3 * 4` Now the `ll` value is '12'.
`return (kk, ll);` It returns the value of variable `ll` only.
The value 12 is stored in variable `'l'`.
Step 4: `printf("%d, %d\n", k, l);` It prints the value of `k` and `l`

Hence the output is "12, 12".

16. What will be the output of the program?

```
#include<stdio.h>
int check(int);
int main()
{
    int i=45, c;
    c = check(i);
    printf("%d\n", c);
    return 0;
}
int check(int ch)
{
    if(ch >= 45)
        return 100;
    else
        return 10;
}
```

A. 100

B. 10

C. 1

D. 0

Answer: Option **A**
Explanation:
Step 1: `int check(int);` This prototype tells the compiler that the function `check()` accepts one integer parameter and returns an integer value.
Step 2: `int l=45, c;` The variable `i` and `c` are declared as an integer type and `i` is initialized to 45.
The function `check(i)` return 100 if the given value of variable `i` is >=(greater than or equal to) 45, else it will return 10.
Step 3: `c = check(i);` becomes `c = check(45);` The function `check()` return 100 and it get stored in the variable `c`.(c = 100)
Step 4: `printf("%d\n", c);` It prints the value of variable `c`.

Hence the output of the program is '100'.

View Answer Discuss in Forum Workspace Report

---

17. If `int` is 2 bytes wide.What will be the output of the program?

```
#include <stdio.h>
void fun(char**);

int main()
{
    char *argv[] = {"ab", "cd", "ef", "gh"};
    fun(argv);
    return 0;
}
void fun(char **p)
{
    char *t;
    t = (p+= sizeof(int))[-1];
    printf("%s\n", t);
}
```

A. ab

B. cd

C. ef

D. gh

Answer: Option **B**
Explanation:
Since C is a machine dependent language `sizeof(int)` may return different values.
The output for the above program will be `cd` in Windows (Turbo C) and `gh` in Linux (GCC).

To understand it better, compile and execute the above program in Windows (with Turbo C compiler) and in Linux (GCC compiler).

View Answer Discuss in Forum Workspace Report

18. What will be the output of the program?

```
#include<stdio.h>
int fun(int(*)());

int main()
{
    fun(main);
    printf("Hi\n");
    return 0;
}
int fun(int (*p)())
{
    printf("Hello ");
    return 0;
}
```

A. Infinite loop

B. Hi

C. Hello Hi

D. Error

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

19. What will be the output of the program?

```
#include<stdio.h>

int fun(int i)
{
    i++;
    return i;
}

int main()
{
    int fun(int);
    int i=3;
    fun(i=fun(fun(i)));
    printf("%d\n", i);
    return 0;
}
```

A. 5

B. 4

C. Error

D. Garbage value

Answer: Option **A**
Explanation:
Step 1: `int fun(int);` This is prototype of function `fun()`. It tells the compiler that the function `fun()` accept one integer parameter and returns an integer value.
Step 2: `int i=3;` The variable `i` is declared as an integer type and initialized to value 3.
Step 3: `fun(i=fun(fun(i)));`. The function `fun(i)` increements the value of `i` by 1(one) and return it.

Lets go step by step,

=> `fun(i)` becomes `fun(3)` is called and it returns 4.
=> `i = fun(fun(i))` becomes `i = fun(4)` is called and it returns 5 and stored in variable `i`.(i=5)
=> `fun(i=fun(fun(i)));` becomes `fun(5);` is called and it return 6 and nowhere the return value is stored.
Step 4: `printf("%d\n", i);` It prints the value of variable i.(5)

Hence the output is '5'.

View Answer Discuss in Forum Workspace Report

---

20. What will be the output of the program?

```
#include<stdio.h>
int fun(int);
int main()
{
    float k=3;
    fun(k=fun(fun(k)));
    printf("%f\n", k);
    return 0;
}
int fun(int i)
{
    i++;
    return i;
}
```

A. 5.000000

B. 3.000000

C. Garbage value

D. 4.000000

Answer: Option **A**

21. What will be the output of the program?

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int i=0;
    i++;
    if(i<=5)
    {
        printf("IndiaBIX");
        exit(1);
        main();
    }
    return 0;
}
```

A. Prints "IndiaBIX" 5 times

B. Function `main()` doesn't calls itself

C. Infinite loop

D. Prints "IndiaBIx"

Answer: Option **D**

Explanation:

Step 1: `int i=0;` The variable `i` is declared as in integer type and initialized to '0'(zero).
Step 2: `i++;` Here variable `i` is increemented by 1. Hence `i` becomes '1'(one).
Step 3: `if(i<=5)` becomes `if(1 <=5)`. Hence the `if` condition is satisfied and it enter into `if` block statements.
Step 4: `printf("IndiaBIX");` It prints "IndiaBIX".
Step 5: `exit(1);` This `exit` statement terminates the program execution.

Hence the output is "IndiaBIx".

1. Point out the error in the program

```
f(int a, int b)
{
    int a;
    a = 20;
    return a;
}
```

A. Missing parenthesis in `return` statement

B. The function should be defined as `int f(int a, int b)`

C. Redeclaration of `a`

D. None of above

Answer: Option **C**
Explanation:
`f(int a, int b)` The variable `a` is declared in the function argument statement.
`int a;` Here again we are declaring the variable `a`. Hence it shows the error "Redeclaration of a"
View Answer Discuss in Forum Workspace Report

---

2. Point out the error in the program

```
#include<stdio.h>
int f(int a)
{
  a > 20? return(10): return(20);
}
int main()
{
    int f(int);
    int b;
    b = f(20);
    printf("%d\n", b);
    return 0;
}
```

A. Error: Prototype declaration

B. No error

C. Error: return statement cannot be used with conditional operators

D. None of above

Answer: Option **C**
Explanation:

In a ternary operator, we cannot use the return statement. The ternary operator requires expressions but not code.

View Answer Discuss in Forum Workspace Report

---

3. Point out the error in the program

```
#include<stdio.h>

int main()
{
    int a=10;
    void f();
    a = f();
    printf("%d\n", a);
```

```
    return 0;
}
void f()
{
    printf("Hi");
}
```

A. Error: Not allowed assignment

B. Error: Doesn't print anything

C. No error

D. None of above

Answer: Option **A**
Explanation:
The function `void f()` is not visible to the compiler while going through main() function. So we have to declare this prototype `void f();` before to main() function. This kind of error will not occur in modern compilers.

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

```
#include<stdio.h>

int main()
{
    printf("%p\n", main());
    return 0;
}
```

A. It prints garbage values infinitely

B. Runs infinitely without printing anything

C. Error: `main()` cannot be called inside `printf()`

D. No Error and print nothing

Answer: Option **B**
Explanation:
In `printf("%p\n", main());` it calls the main() function and then it repeats infinetly, untill stack overflow.
View Answer Discuss in Forum Workspace Report

---

2. There is a error in the below program. Which statement will you add to remove it?

```
#include<stdio.h>

int main()
{
```

```
    int a;
    a = f(10, 3.14);
    printf("%d\n", a);
    return 0;
}
float f(int aa, float bb)
{
    return ((float)aa + bb);
}
```

A. Add prototype: `float f(aa, bb)`

B. Add prototype: `float f(int, float)`

C. Add prototype: `float f(float, int)`

D. Add prototype: `float f(bb, aa)`

Answer: Option **B**
Explanation:
The correct form of function `f` prototype is `float f(int, float);`
View Answer Discuss in Forum Workspace Report

---

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

```
long fun(int num)
{
    int i;
    long f=1;
    for(i=1; i<=num; i++)
        f = f * i;
    return f;
}
```

A. The function calculates the value of 1 raised to power num.

B. The function calculates the square root of an integer

C. The function calculates the factorial value of an integer

D. None of above

Answer: Option **C**
Explanation:
Yes, this function calculates and return the factorial value of an given integer `num`.