Quiz-6

Question 1

Time: 00:00:07

What will be the output of the following code?

```
#include <iostream>

using namespace std;

int main ()

{

  int x = 0, y = 1;

  for (; y; cout << ++x << y++ << " ")

    {

      x = y++ <= 15;

    }

  return 0;

}
```

○ 31,71,41,60,21	11.86%	○ 22 23 44 55	16.72%
○ 21 31 51 61	14.81%	◉ Infinite loop	56.60%

The code will get into an infinite loop after entering in the for loop, and it will keep printing the value of y

Question 2

Time: 00:00:03

What will be the output of the following code?

```
#include<stdio.h>

int main ()

{

  int i = 1, j;

  for (;;)

    {

        i++;

      if (i)

        j = --i;

      if (i < 5)

        printf ("Royal Pass ", i++);

      else

        break;

    }

  return 0;

}
```

- will print Royal Pass 4 times
- No, compile error but it will run into an infinite loop printing Royal Pass.
- No, compile error but it'll not print Royal Pass.
- Compile-time error.

In every iteration the value of i is getting incremented by 1, (we can cancel the operation --i, with ++i, as these operations are there just for creating confusion). So initially the value of i=1, it will get incremented every time and print Royal Pass until the if condition gets false i.e; the value becomes 5, hence this code will print Royal Pass 4 times

Question 3

Time: 00:00:03

What will be the output of the following code?

```
#include<stdio.h>

int main ()

{

  int x = 4, y = 0;

  int z;

  z = (x++ + ++y + y++ , x++);

  printf ("%d\n", z);

  return 0;

}
```

- 5
- zero - '0'
- compiler error
- undefine behavior due to the order of evolution can be different

The code is free of errors in the line where

z = (x++ + ++y + y++ , x++);

Here z will get the value of the last expression of this set.
from the first expression which is x++ + ++y + y++ the value of x will be 5.
As, x++ is post increment hence z will get 5 first then x will be incremented. So, the answer will be 5.

Question 4

Time: 00:00:10

What will be the output of the following code?

```
#include <stdio.h>

int f(int n)

{

  if (n==0)

  return 1;

  else

  return f(n-1);

}


int main ()

{

  printf ("Result:%d", f (50));

  return 0;

}
```

- Run time error 8.06%
- Compile time error 10.63%
- Result:0 1 2 3 4 .... 50 21.33%
- **Result:1 59.98%**

The code is a basic application of recursion technique which prints a user required result, the function f(int n), will keep on calling itself until the value of n becomes 0, and then finally it will print 1

Question 5

Time: 00:00:01

What will be the output of the following C code?

```
#include <stdio.h>

int f(int n)

{

  if (n==0)

  return 1;

  else

  return n+f(n-1);

}


int main ()

{

  printf ("%d", f (10));

  return 0;

}
```

- compile time error
- infinite loop
- 56
- 55

The code is an example of recursion, where the function f(int n), call itself again and again until the value of n becomes zero, and on every call it is adding up the values of n, i.e; the function is adding 10+9+8+7+6+5+4+3+2+1+1 = 56, the extra 1 will be added when the value of n becomes zero, hence the output will be 56

Question 6

Time: 00:00:33

What will be the output of the following C code?

```
#include <stdio.h>

int main ()

{

  int x = 5, y = 4, z = 3;

  int a = x<<+1;

  int b= y>>+1;

  int c = z<<+1;

  printf ("%d\n", a);

  printf ("%d\n", b);

  printf ("%d\n", c);



  return 0;

}
```

- Error
- 10
  4
  2
- 10
  2
  6
- 10
  2
  4

The following code is performing bitwise shift operations on the value of x, y and z
So, a becomes x<<+1=5*2=10
b becomes y>>+1=4/2=2
c becomes z<<+1=3*2=6
Hence the answer is
10
2
6

Question 7

Time: 00:00:01

What will be the output of the following C code? [assume it is a 64 bit-computer]

```
#include <stdio.h>

struct node

{

    int data;

    int *pointer;

};

int main ()

{

  node m;

  printf ("%d", sizeof (m));

  return 0;

}
```

- 4
- 12
- 16
- 5

This happens due to Structural Padding. In a 64 bit computer, the frames take 8 bytes, and the variables are stored in frames. That's why if you take another integer variable after the first one, the total memory of the structure will remain the same.
In our case the data variable stays in a 8 bytes memory and pointer in stays in another 8 byte memory.
Hence 16 byte is the answer.

Question 8

Time: 00:00:00

What will be the output for the pseudocode for x=10, y=25

```
fun(int x,int y)
    if(x==0)
        return y;
    else
        return fun(x-1 , y+1)
```

- 25
- 9
- 6
- None of the above

```
int fun (int x, int y)
{
    if (x == 0)
  return y;
else
  return fun (x-1, y+1);
}
int main()
{
    int y=10;
    int x=25;
   printf("%d",fun(x,y));
}
```

The above code uses the recursion of a function to reach the final output. On every function call, the value of x is decreasing by 1 and the value of y is being updated as 'y+1'. So, in the final call the value of x=0 and y=35, this will be executed and 35 will be printed as the final output.

Question 9

Time: 00:00:01

What will be the output of the following code?

```
#include<stdio.h>

int main ()

{

  int no = 1112, temp, digit, sum = 1;

  temp = no;

  while (no > 0)

    {

      digit = no % 10;

      sum = sum * digit;

      no /= 10;

    }

  printf ("%d\n", sum);

  return 0;

}
```

- ○ 25
- ○ 7
- ○ 5
- ○ 2

The above code is doing the multiplication of digits of the given number, i.e., 1 x 1 x1 x 2 = 2, which is the output of the given code.

Question 10

Time: 00:00:01

What will be the output of the following C code

```
#include<stdio.h>

int main ()

{

  int x = 21, y = 10, z = 13;

  x+y>5 ? (printf ("%d", z)) : (return z);

}
```

- error
- 10
- 2
- 13

expr1?expr2:expr3 works as follows: expr1 is evaluated first if it is true, expr2 is evaluated, otherwise expr3 is evaluated. Hence in expressions the **return** statement can not be used in **C**-language. The **return** statement is used for **returning** from a **function** , you can't **use** inside **ternary operator**

Pseudocode - 6

1) Steps:-

i) initialise e.g. x = 0, y = 1;

ii) for loop structure:
    No initialization
    Condition: y is non-zero
    cout << ++x << y++ << "\n"

iii) ++x is invalid, there are no variable xx declared. which cause compilation error.
issues. Compilation error & logical error.
4. Hypothetical output (if ++x where ++x)

output:- Contains compilation error due to undefined variable.

2) i) initialization:- i = 1, j is uninitialized

ii) infinite loop (for(;;)).

It runs indefinitely, unless Break statement

iii) First iteration (i = 1).

i++ → i = 2
if (i) is true → j = --i → 1 = 1, j = 1

COMPASS

M T W T F S S

Date:

- if (i<5) is true (1<5), print "Royal pals" & then i++ → i=2

iv) Second iteration (i=2):

i++ → i=3
if (i) is true (i=3), so j=--i → i=2, j=2
if (i<5) is true (2<5) print "Royal pals" & j++ → i=3.

v) Third iteration (i=3):

i++ → i=4.
if (i) is true (i=4), so j=--i → i=3, j=3
if (i<5) is true (3<5) print "Royal pals" & i++ → i=4.

vi) Fourth iteration (i=4):

i++ → i=5.
if (i) is true (i=5), so j=--i → i=4, j=4
if (i<5) is true (4<5) print "Royal pals" & i++ → i=5

vii) Fifth iteration (i=5):
i++ → i=6
if (i) is true (i=6) so i=--i → i=5
if (i<5) is false (5<5), so break exit the loop.

M T W T F S S

COMPASS

Date :

Output :- "Royalpass" print 4 times for i = 1, 2, 3, 4.

3) i) initialization: x = 4, y = 0

ii) Expression :- (x++ + ++x + y++, x)

iii) x++: x returns 4, then x = 5

++xy: Should be ++x or ++y

++x + y => +4 + 0 = 0

++x + y => +4 + 0 = 4

y++: y returns 0, then y = 1

Ans:- z = 5 + 1 = 6

Output :- Code contain compile time error due to invalid expression ++xy. (Compile error)

4) i) f(int n):

if n == 0, return ; (Base case)

Recursive call : f(n-1) until n == 0.

ii) Call f(50) it recursively called f(49), f(48) ... f(0)

when n == 0, fn should return ; doesn't exist

iii) Code will not compile due to undefined variable i. (Compile time error)

COMPASS

M T W T F S S

Date:

5) Step:-

(1) F(int n):
if n==0, return 1 (Base case)
Critical error: compilation error.

- Recursive case: n * f(n-1).
Critical error: i is undefined here.

(2) Call f(10), 10 * f(10-i)
i is undefined, the code can't compile.

(3) possible:- i=1, factorial of 10
i=2, product of all nos.

- Original code is invalid due to declaration of 1.

4) - int data: size = 4 bytes
- int * pointer: size = 8 bytes.

- Total size of struct node = sizeof(int data) + sizeof(int * pointer) = 4+8 = 12 bytes

- print(12) (printf("%d", size of))

6) Step:-

1. Initialization:
x=5, y=4, z=3
2. Bitwise operations:

a = x << 1;
left shift 5 by 1 bit: 5 in binary is 101
→ shifted left by 1 give 1010 = 10

- b = y >> 1;
  Right shift 4 by 1 unit: 4 in binary is 100 → shift right by 1 give 10 binary = 2 decimal
- c = 3 << 1;
  left shift 3 by 1 bit: 3 in binary is 11 shift left by 1 give 110 binary = 6 decimal

print (10)
print (2)
print (6)

8) Steps:-
i) fun(25, 10) → calls fn(24, 11)
fun(24, 11) → call fun(23, 12)
fun(1, 34) → call fun(0, 35)
fun(0, 35) → return 35 (base case).

1. final y = initial y + initial x

initial x = 25
" y = 10
final result = 10 + 25 = 35

9) i) initial value: n = 112, Sum = 1

ii) First iteration:- digit = 112 % 10 = 2
Sum = 1 + 2 = 3

no = 1112 / 10 = 111

Second iteration :

digit = 111 % 10 = 1
Sum = 2 * 1 = 2
no = 111 / 10 = 11

Third iteration :-

digit = 11 % 10 = 1
Sum = 2 * 1 = 2
no = 11 / 10 = 1

Fourth iteration :- digit = 1 % 10 = 1
Sum = 2 * 1 = 2
no = 1 / 10 = 10 //

Output :- program print 2.