C Aptitude Questions

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

Output:

2 1
3 1
4 1
5 1
6 1
7 1

Explanation:

To understand this question firstly we need to understand how for loop works in c. Below is the syntax of the for loop.

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

All the three statements are optional here. In our question the first statement is ignored.
Now the statement 2 is executed each time the loop statement begins.
Statement 3 is not executed for the first time.
So tracing the program will result in the output

Iteration 1

x, y = 1, 1 // j => 1(True)
y = x++ <= 5; => 1 <= 5 True(all integers except 0 in our case it is 1)
x = 2

Now statement 3 is executed so outputs 2 1
Now statement 2 condition is checked y -> 1 (True)

Iteration 2
x, y = 2, 1
y = x++ <= 5; -> 2 <= 5 -> 1
x = 3
Now statement 3 is executed so outputs 3 1
Now statement 2 condition is checked y -> 1 (True)

Iteration 3

x, y = 3, 1
y = x++ <= 5; -> 3 <= 5 -> 1
x = 4
Now statement 3 is executed so outputs 4 1
Now statement 2 condition is checked y -> 1 (True)

Iteration 4

x, y = 4, 1
y = x++ <= 5; -> 4 <= 5 -> 1
x = 5
Now statement 3 is executed so outputs 5 1
Now statement 2 condition is checked y -> 1 (True)

Iteration 5

x, y = 5, 1
y = x++ <= 5; -> 5 <= 5 -> 1
x = 6
Now statement 3 is executed so outputs 6 1
Now statement 2 condition is checked y -> 1 (True)

Iteration 6

x, y = 6, 1
y = x++ <= 5; -> 6 <= 5 -> 0(False)
x = 7
Now statement 3 is executed so outputs 7 0
Now statement 2 condition is checked y -> 0 (False)

Loop terminates.

2.

```
#include<stdio.h>

int main(){
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
```

```
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

Output:

3, 2, 15

Explanation:

Tracing the program i = ++a[1] -> i = 2 and a[1] = 2

j = a[1]++ -> j = 2 and a[1] = 3

m = a[i++] -> m = a[2] = 15 and i = 3

printing this will result in the output

3.

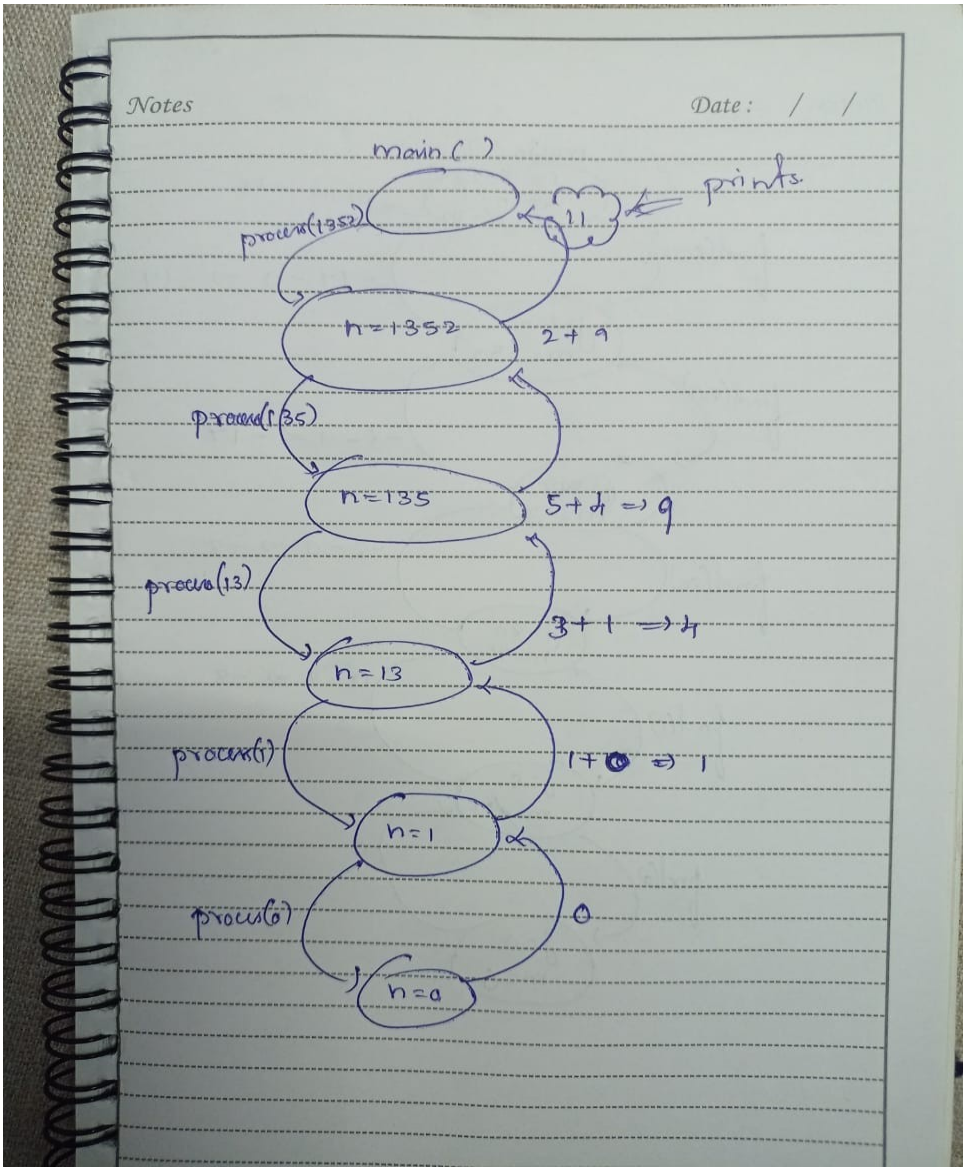
```
int process(int n){
        return n == 0 ? 0 : n % 10 + process(n / 10);
}

int main(void){
    printf("%d", process(1352));
    getchar();
    return 0;
}
```

Output:

11

Explanation:

The following tracing of the recursive function calls will help you understand the output.



4.

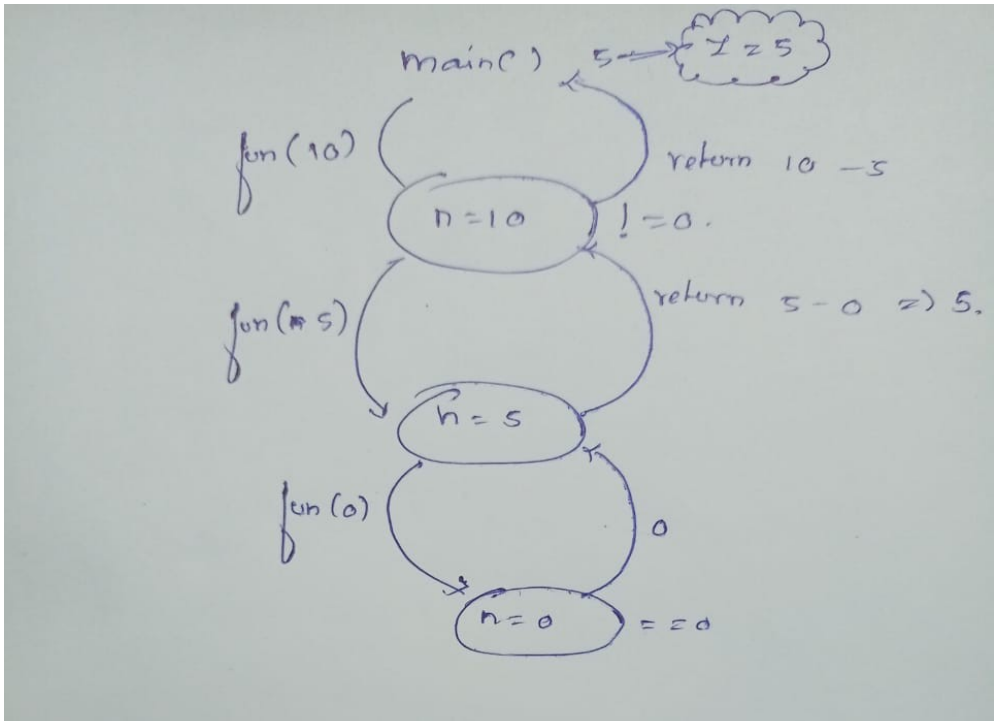
```
int fun(int n){

    if(n != 0)
        return n - fun(n - 5);
    else
        return n;

}

int main(){
    int n = 10, z;
    z = fun(n);
    printf("%d", z);
}
```

Output:

5

Explanation:

The following tracing of the recursive function calls will help you understand the output.



5.

```
#include<stdio.h>

int min(int x, int y){
    return (y < x) ? y : x;
}

int main(){
    int a[] = {-5, 9, 8, -8, -2};
    int z = a[0], n = 5, i = 0, c = a[0];
    for(i = 1; i < n; i++){
        c = min(a[i], c + a[i]);
        z = min(z, c);
    }
    printf("%d", z);
}
```

Output
-10

Explanation

Iteration 1:

z = -5, n = 5, i = 1, c = -5
c = min(a[i], c + a[i]); -> min(9, 4) = 4
z = min(z, c); -> min(-5, 4) = -5

Iteration 2:

z = -5, n = 5, i = 2, c = 4
c = min(a[i], c + a[i]); -> min(8, 12) = 8
z = min(z, c); -> min(-5, 8) = -5

Iteration 3:

z = -5, n = 5, i = 3, c = 8
c = min(a[i], c + a[i]); -> min(-8, 0) -> -8
z = min(z, c) -> min(-5, -8) -> -8

Iteration 4:

z = -8, n = 5, i = 4, c = -8
c = min(a[i], c + c[i]); -> min(-2, -10) -> -10
z = min(-8, -10) -> -10

So it prints an output -10

6.

```
#include<stdio.h>

int main(){
    int a = 10, b = 20, c = 30;
    if(c > b > a)
        printf("True");
    else
        printf("False");
}
```

Output:

False

Explanation:

first the conditional statements have a left to right associativity so on evaluating (c > b > a) -> (30 > 20 > 10) -> (1{True} > 10) -> (False) so the if block fails and the else block gets executed.

7.

```
#include<stdio.h>

int main(){

    int c[] = {5, 0, 3, 4, 5};
    int j, *q = c;

    for(j = 0; j < *q; j++){
        printf("%d ", *c);
        ++q;
    }

    return 0;
}
```

Output:

5

Explanation:

Iteration 1:

q = &c[0](&c), j = 0, j < *q -> 0 < 5 -> True
prints *c -> c[0] -> 5
++q -> q = q[1]

Iteration 2:

j = 1, *q = 0 -> j < *q -> 1 < 0 False:
loop terminates so the output is only 5

8.

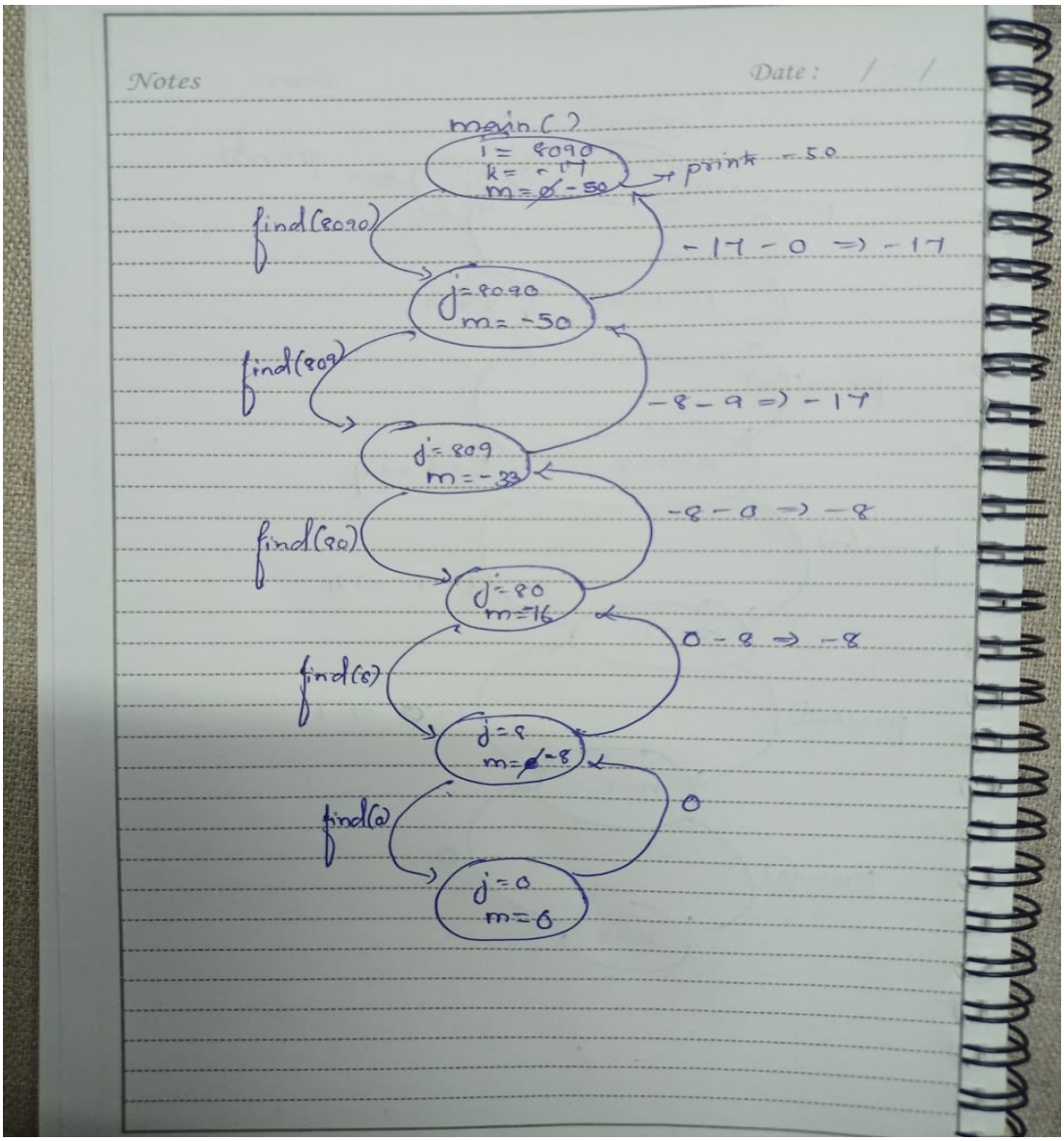
```
#include<stdio.h>
int m = 0;
int find(int j){

    if(j > 1){
        j = find(j / 10) - (j % 10);
        m += j;
    }
    else{
        j = 0;
    }

    return j;

}


int main(){

    int i = 8090;
    int k;
    k = find(i);
    printf("%d ", m);

    return 0;

}
```

Output:

-50

Explanation:

The following tracing of the recursive function calls will help you understand the output.



9.

```
#include<stdio.h>

void main(){
   int n = 11, res = 1;

   do{
      n -= 5;
      res *= 5;
   }while(n > 5);

   printf("%d", n * res);
}
```

Output:

25

Explanation:

Iteration 1:

n = 11 -> 6
res = 1 -> 5

Iteration 2: (6 > 5) True
n = 6 -> 1
res = 5 -> 25

Iteration 3: (1 > 5) False Terminates

prints the value of res...

10.

```
#include<stdio.h>

void function(int[][3]);

int main(void){
    int a[3][3] = {1, 2, 3, 4, 5, 6, 7, 8 , 9};
    function(a);
    printf("%d", a[2][1] - a[1][2]);

    return 0;
}

void function(int a[][3]){

    ++a;
    a[1][1]++;
}
```

Output:

3

Explanation:

Generally a would point to the a[0][0] ‘s reference
now calling the function function(a);

This passes the reference of the first element ie., a[0][0]
now ++a ->a = &a + (row + 1){for simplicity i have added like this}
so a would point to the second row in the array

a[1][1] ++ would mean the third row from the original array and col 2 so the element 8 is incremented to 9

then the function returns and then
a[2][1] – a[1][2] -> 9 – 6 -> 3(Output)

Note both the a in main and function are different both of them points to different rows in certain scenarios