1.
```
void main(){
   int i = 4, j = 8;
   i = i|j & j|i + i|j & j|i - i^j;
   j = i|i & j|j + j|j & i|i - j^j;
   printf("%d %d %d\n", i|j & j|i, i|j &j|i, i^j);
}
```

The question is related to operator precidence....
See the following operator precidence table

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= ;=	Right to left
Comma	,	Left to right

The operations to be performed is of the order
* Addition/Subraction
* Bitwise AND
* Bitwise XOR
* Bitwise OR

so the expression

i = i|j & j|i + i|j & j|i – i^j; -> i = 4|8 & 8|4 + 4|8 & 8|4 – 4^8; -> 4|8 & 8| 8 |8 & 8| 0 ^8; -> 4 | 8 | 8 | 8 | 0 ^8

-> 4 | 8 | 8 | 8 | 8 -> 4 | 8 -> 12

so the updated value of i would be 12

next expression

j = i|i & j|j + j|j & i|i – j^j; -> 12|12 & 8|8 + 8|8 & 12|12 – 8^8 -> 12|12 & 8| 16 |8 & 12| 4 ^8 ->

12| 8 | 16 | 8 | 4 ^8 -> 12| 8 | 16 | 8 | 12 -> 12 | 8 | 16 -> 28

So the updated value of j would be 28

lastly the expression in the print statements

i|j & j|i -> 12|28 & 28|12 -> 12| 28 | 12 -> 12|28 -> 28

i^j -> 12 ^ 28 -> 16

So the output would be
28 28 16

2.
```
void main(){
   int b = 9093;
   int a = 0;
   while(b > 0){
      a = a + (b % 10);
      b = b/10;
   }
   printf("%d\n", a);
}
```

The initial values of a and b are 9093 and 0 respectively

Iteration 1 -> begin -> a = 0, b = 9093, end -> a = 0 + 3 = 3, b = 9093 / 10 = 909
Iteration 2 -> begin -> a = 3, b = 909, end -> a = 3 + 9 = 12, b = 909 / 10 = 90
Iteration 3 -> begin -> a = 12, b = 90, end -> a = 12 + 0,= 12, b = 90/10 = 9
Iteration 4 -> begin -> a = 12, b = 9, end -> a = 12 + 9 = 21, b = 9/10 = 0

`b` has became 0 so there wont be a next iteration so the final value of a would be `21`

3.
```
void main(){
   int a[3][4] = {2, 4, 6, 8, 10, 12, 12, 10, 8, 6, 4, 2};
   int i = 0, j, k = 13;
   while(i < 3){
      for(j =0; j < 4; j++){
         if(a[i][j] > k)
            k = a[i][j];
      }
      i++;
   }
   printf("%d\n", k);
}
```

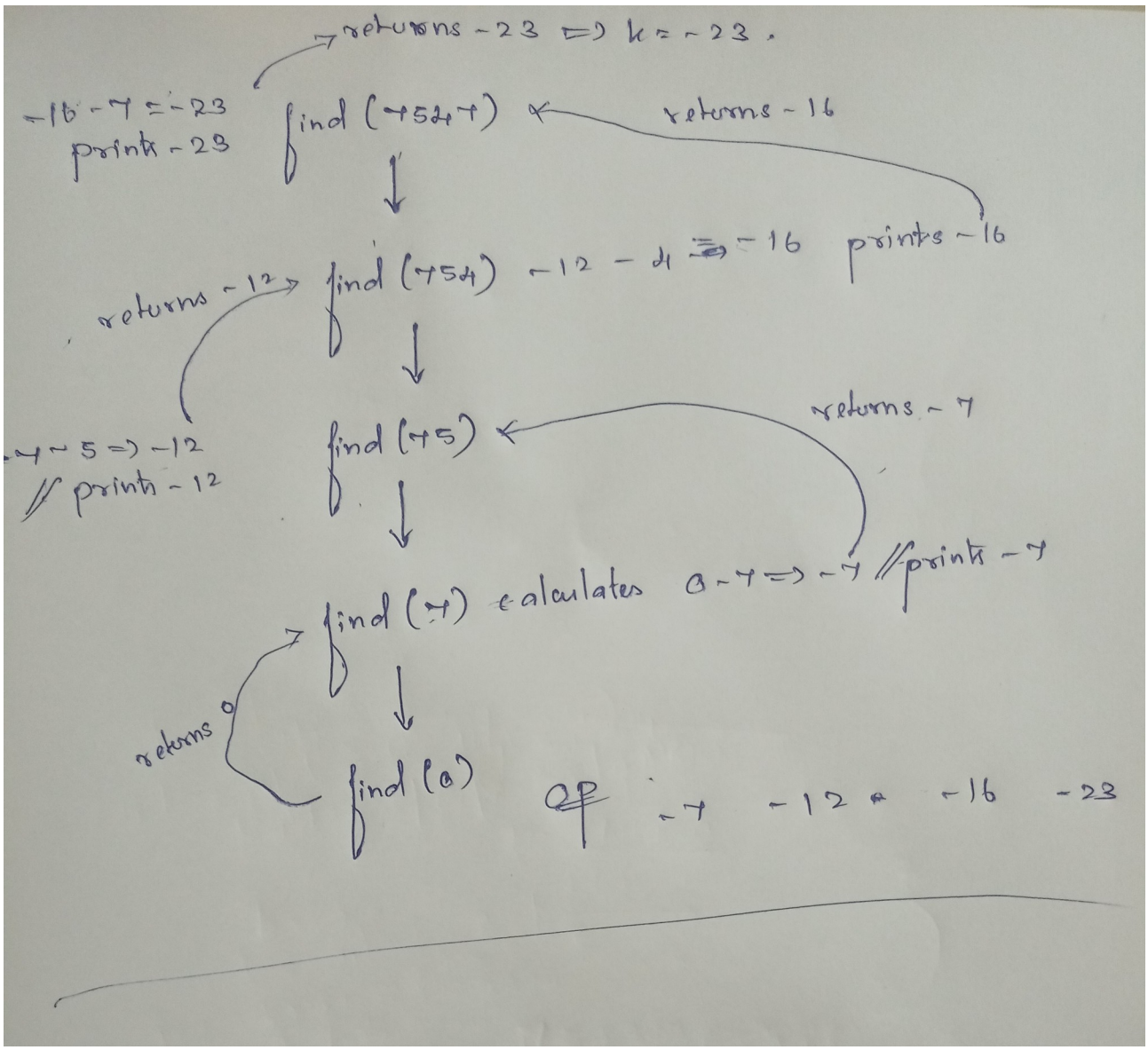
As the value of k is greater than the values of all elements in the matrix `a` the if block is not executed so the value of k would be 13 and it does not change

4.
```
int find(int j){
   if(j > 1){
      j = find(j / 10) - (j % 10);
      printf("%d\t",j);
   }
   else{
      j = 0;
   }
   return j;
}


void main(){
   int i = 7547;
   int k;
   k = find(i);
}
```

Solution:

Forming the recurssive tree....



The Output would be
-7 -12 -16 -23

```
5. int a[] = {2, 4, 6};
int *f(void){
    int i;
    for(i = 0; i<3; i++)
        return a+1;
}

void main(){
    *f() = 12;
    printf("%d %d %d", a[0], a[1], a[2]);
}
```

Solution :
The program begins its execution from the main()

The first line to be executed is *f() = 12;

This makes a function call to the function f with return type of an integer pointer

The function returns the reference of a + 1 {a means the a[0]th reference so a + 1 means a[1]’s reference}

so the first line becomes *(a + 1) = 12 -> a[1] = 12

next the printf statement prints -> 2 12 6

6.
```
void main(){
   int n;
   for(n = 5; n > 0; n--){
      printf("%d", n--);
   }
}
```

Solution:

since the decrement is postfix the value of i is printed first and then i is decremented

Iteration 1 -> i = 5 (Checks i != -1 True) -> outputs 5 and then i = 4
Iteration 2 -> i-- -> i = 3 (Checks i != -1 True) -> outputs 3 and then i = 2
Iteration 3 -> i-- -> i = 1 (Checks i != -1 True) -> outputs 1 and then i = 0
Iteration 4 > i-- -> i = -1 (Checks i != -1 True) -> outputs -1 and then i = -2

The program executes infinately....

7.
```
void main(){
   int c[] = {1, 2, 3, 4};
   int j, *q = c;
   for(j = 0; j<4;j++){
      printf("%d", *c);
      ++q;
   }
}
```

The output is
1111

Explanation:

This is something that we should understand about continious memory allocation in c
The elements in the array are allocated continiously....
So the address of the first element is stored in the variable c
dereferencing it gives the value of the first element in the array

bonus point : if you are trying to dereferrence *(c + 1) -> this means *(c + 1 *(sizeof(int))

for more information see memory allocation of arays in c.

8.
```
#include<string.h>

void main(){
   int nf, i, j, c, m;
   char str[] = {"Zoho Corporation - Chennai"};
   int length = strlen(str);
   i = 0, m = 0, c = 0;

   while(str[i] != '\0' && i <= length ){
      j = i;
      c = 0;
      while(str[j] != ' ' && str[j] != '\0'){
         j++;
      }
      i = j+1;
      while(--j && str[j] != ' '){
         if(str[j] == 'o'){
            c++;
         }
      }
      if(m < c)
```

```
            m = c;
    }
    printf("%d", m);
}
```

Explanation:

`o`This is a program used to count the no of `0` from a word which has the maximum `o` so the ans would be from the word Corporation which has 3 `o`s

So the output is 3