Predict the output of following C programs.

```c
// PROGRAM 1
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    ++*p;
    printf("arr[0] = %d, arr[1] = %d, *p = %d", arr[0], arr[1], *p);
    return 0;
}
```

```c
// PROGRAM 2
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    *p++;
    printf("arr[0] = %d, arr[1] = %d, *p = %d", arr[0], arr[1], *p);
    return 0;
}
```

```c
// PROGRAM 3
#include <stdio.h>
int main(void)
{
    int arr[] = {10, 20};
    int *p = arr;
    *++p;
    printf("arr[0] = %d, arr[1] = %d, *p = %d", arr[0], arr[1], *p);
    return 0;
}
```

The output of above programs and all such programs can be easily guessed by remembering following simple rules about postfix ++, prefix ++ and * (dereference) operators

1) Precedence of prefix ++ and * is same. Associativity of both is right to left.

2) Precedence of postfix ++ is higher than both * and prefix ++. Associativity of postfix ++ is left to right.

The expression **++*p** has two operators of same precedence, so compiler looks for assoiativity. Associativity of operators is right to left. Therefore the expression is treated as ***++(*p)***. Therefore the output of first program is "*arr[0] = 11, arr[1] = 20, *p = 11*".

The expression ***p++** is treated as ****(p++)*** as the precedence of postfix ++ is higher than *. Therefore the output of second program is "*arr[0] = 10, arr[1] = 20, *p = 20*".

The expression ***++p** has two operators of same precedence, so compiler looks for assoiativity. Associativity of operators is right to left. Therefore the expression is treated as ****(++p)***. Therefore the output of second program is "*arr[0] = 10, arr[1] = 20, *p = 20*".